LA CRITTOGRAFIA. Storia e scienza del segreto

Questo contributo nasce da una duplice esigenza. Da un lato, la crittografia rappresenta una delle discipline più affascinanti e ricche di implicazioni che l'umanità abbia sviluppato - un territorio dove la matematica incontra la storia, la linguistica si intreccia con l'ingegneria, e la teoria astratta trova applicazioni che hanno cambiato il corso degli eventi mondiali. Dall'altro lato, proprio questa ricchezza rende la crittografia una materia che può essere affrontata a diversi livelli di profondità, e che merita approcci complementari per essere pienamente compresa.

Il testo che segue si propone come un viaggio narrativo attraverso quattromila anni di evoluzione crittografica, dove la storia non è semplice cornice ma parte integrante della comprensione. Vedremo come i sistemi crittografici non nascono nel vuoto ma rispondono a necessità concrete - militari, diplomatiche, commerciali - e come ogni innovazione tecnica sia inscindibile dal suo contesto storico. Al contempo, manterremo il rigore necessario per comprendere i principi matematici sottostanti, perché è proprio in questi principi che risiede l'universalità della crittografia, la sua capacità di trascendere le epoche e le culture.

Accompagna il presente paper un trattato tecnico più esteso, pensato per chi desideri approfondire gli aspetti formali, le dimostrazioni complete, gli esempi numerici dettagliati e le analisi matematiche più raffinate. I due testi sono concepiti come complementari: questo saggio offre la narrazione continua e l'intuizione dei concetti; il trattato fornisce l'apparato formale e la sistematizzazione completa. Il lettore potrà scegliere di seguire solo questa narrazione per avere una comprensione panoramica ma sostanziale, oppure alternare la lettura del saggio con le sezioni corrispondenti del trattato tecnico per un'immersione più profonda nella matematica.

La scelta di questo approccio bipartito nasce dalla convinzione che la crittografia sia troppo importante per rimanere confinata agli specialisti, ma troppo profonda per essere banalizzata. C'è una via di mezzo tra il divulgativo superficiale e il tecnico impenetrabile, ed è in quello spazio che questo saggio vuole collocarsi - rigoroso senza essere arido, accessibile senza essere semplicistico, narrativo senza rinunciare alla precisione.

Il percorso che ci attende inizia nelle pianure della Mesopotamia, circa quattromila anni fa, e ci condurrà fino alle frontiere della crittografia quantistica e post-quantistica. Vedremo nascere i primi sistemi di oscuramento informativo, assisteremo alla rivoluzione concettuale di al-Kindī che trasformò la crittanalisi in scienza, seguiremo la corsa agli armamenti intellettuali del Rinascimento, vivremo il dramma di Enigma e Bletchley Park, e infine esploreremo come la crittografia moderna stia affrontando la minaccia dei computer quantistici.

Attraverso questo viaggio emergerà un tema ricorrente: il rapporto dialettico tra cifratura e decifratura, tra chi cerca di proteggere i segreti e chi cerca di violarli. Questa tensione dinamica ha guidato l'evoluzione della crittografia per millenni,

producendo innovazioni che spesso hanno anticipato sviluppi matematici più generali. La teoria dei gruppi, l'algebra astratta, la teoria dell'informazione, la complessità computazionale - tutte queste discipline hanno trovato nella crittografia terreno fertile per applicazioni e, talvolta, motivazioni originarie.
Un'ultima considerazione metodologica. Nel raccontare questa storia, ho scelto di privilegiare la continuità concettuale rispetto alla completezza enciclopedica. Alcuni sistemi crittografici storicamente rilevanti riceveranno trattazione dettagliata perché incarnano principi fondamentali o rappresentano salti qualitativi; altri, pur importanti, saranno menzionati più brevemente se non aggiungono intuizioni essenziali alla narrazione. L'obiettivo non è catalogare ogni cifrario mai inventato, ma tracciare la linea evolutiva delle idee, mostrare come i concetti si sviluppino, si raffinino, e talvolta vengano risoperti in contesti nuovi.
Iniziamo dunque il nostro viaggio, consapevoli che la crittografia non è solo una disciplina tecnica ma un modo di pensare - un'arte del nascondere che ci insegna, paradossalmente, molto su come funziona la comunicazione, su cosa significhi conoscere, e su come la matematica possa essere non solo descrittiva ma anche protettiva, non solo strumento di comprensione ma anche di difesa.

Le Origini (3000 a.C. - 400 d.C.).

L'Oscuramento come Protezione Sociale

Quando gli scribi mesopotamici del II millennio avanti Cristo alteravano intenzionalmente i loro testi per proteggere le formule degli smalti vetrosi, non stavano facendo crittografia nel senso moderno del termine. Non esistevano ancora i concetti di chiave, di algoritmo reversibile, di separazione tra metodo e segreto. Eppure, in quei gesti - la sostituzione di una forma grafica con un'altra più antica, la decomposizione di un logogramma in sillabe, l'omissione strategica dei determinativi - possiamo riconoscere la prima manifestazione consapevole di un'intuizione destinata a attraversare i millenni: l'informazione ha valore, e chi la possiede può volerla proteggere.
Il sistema cuneiforme, con la sua straordinaria complessità stratificata dove ogni segno poteva funzionare come ideogramma, fonogramma o classificatore semantico, offriva un terreno naturalmente fertile per l'oscuramento. La protezione non risiedeva in una trasformazione matematica ma in una barriera sociale: solo chi apparteneva all'élite scribale più istruita, solo chi conosceva le varianti arcaiche e le convenzioni esoteriche, poteva accedere ai segreti. Era una crittografia sociologica piuttosto che algoritmica, ma non per questo meno efficace nel suo contesto. Un artigiano privo dell'educazione scribale non avrebbe potuto decifrare quelle formule anche avendole sotto gli occhi.
Gli astronomi babilonesi, nel periodo neo-babilonese, fecero un passo ulteriore. I loro coefficienti astronomici - come il periodo sinodico lunare, fondamentale per calendari

e predizioni delle eclissi - venivano codificati non semplicemente attraverso varianti grafiche ma attraverso espressioni aritmetiche deliberatamente complesse. Un coefficiente veniva presentato come risultato di operazioni che richiedevano conoscenza contestuale per essere interpretate correttamente. Qui intravediamo qualcosa di più vicino alla crittografia moderna: una trasformazione intenzionale dei dati che, per essere invertita, richiede più della semplice alfabetizzazione - richiede la "chiave" della conoscenza astronomica specialistica.
Nell'Antico Egitto, il sistema geroglifico offriva possibilità analoghe di oscuramento attraverso la complessità semiotica. Ma l'analisi delle iscrizioni tombali - come quella celebre di Menet-Khufu a Beni Hassan - rivela che l'obiettivo primario non era tanto rendere il testo illeggibile quanto conferirgli prestigio, sacralità, protezione magica. Le forme arcaiche dell'Antico Regno, la moltiplicazione grafica dei segni, l'elaborazione decorativa erano gesti carichi di significato culturale e religioso più che puramente funzionali alla segretezza. È un promemoria importante: non tutto ciò che è oscuro è crittografia, e non ogni alterazione dei segni risponde alla stessa logica.
Nel periodo tardo, tuttavia, con l'influenza greca e lo sviluppo della gematria copta - dove ogni lettera riceve un valore numerico fisso e le parole possono essere sostituite dalle loro somme - ci avviciniamo a una vera crittografia per sostituzione. Qui c'è una funzione definita, una trasformazione reversibile, l'intenzione esplicita di nascondere significati. Certo, la funzione non è iniettiva - molte parole diverse possono avere lo stesso valore numerico - ma il principio è stabilito: il linguaggio può essere sistematicamente trasformato in altro (numeri, in questo caso) per proteggere i contenuti.

Le Prime Trasformazioni Algoritmiche

Lo scitale spartano rappresenta un salto concettuale decisivo. Per la prima volta nella documentazione storica che possediamo, troviamo un dispositivo - due bastoni cilindrici identici nel diametro - che implementa una trasformazione puramente algoritmica. La logica è elegante nella sua semplicità: avvolgendo una striscia di cuoio attorno al cilindro e scrivendo il messaggio longitudinalmente, poi srotolando la striscia, i caratteri si trovano dispersi secondo un ordine che dipende esclusivamente dalla geometria del cilindro. Non c'è oscuramento simbolico, non c'è sostituzione - solo riorganizzazione posizionale.
Matematicamente, lo scitale implementa una permutazione di trasposizione. Se immaginiamo il messaggio disposto in una matrice rettangolare, la cifratura consiste nel leggere per colonne ciò che è stato scritto per righe. La funzione è biunivoca, quindi perfettamente reversibile. E qui emerge un concetto fondamentale: il diametro del cilindro è la chiave - un parametro separabile dal metodo. Chiunque comprenda il principio dello scitale può tentare di decifrare il messaggio, ma senza il cilindro del diametro corretto (o la capacità di testare sistematicamente tutti i diametri possibili), il successo rimane improbabile.

Lo scitale ha caratteristiche che lo rendono emblematico. Da un lato, lo spazio delle chiavi è sorprendentemente piccolo - le limitazioni manifatturiere dell'antichità significano che esistono forse solo sessanta diametri plausibili distinti - rendendo il sistema vulnerabile a un attacco esaustivo anche con mezzi antichi. Dall'altro, la trasposizione pura preserva completamente le frequenze dei caratteri, rendendo l'analisi di frequenza (che al-Kindī sistematizzerà secoli dopo) completamente inefficace. La vulnerabilità dello scitale sta altrove: nei pattern dei digrammi, nelle sequenze di coppie di lettere che mantengono relazioni posizionali riconoscibili.
L'episodio riportato da Plutarco - Lisandro che riceve via scitale l'ordine di richiamo mentre opera in Asia Minore, messaggero intercettato dai Persiani che però non riescono a estrarre significato dalla striscia di cuoio - illustra perfettamente sia la forza che i limiti del sistema. Funziona in condizioni operative dove l'attaccante ha tempo limitato, dove non sono disponibili cribs (contenuti ipotizzabili), dove il volume del materiale intercettato è insufficiente per analisi sistematiche. Ma in laboratorio, con tempo e testi multipli, lo scitale sarebbe vulnerabile.

La Sostituzione Sistematica e la Sua Fragilità

Il cifrario di Cesare, testimoniato da Svetonio circa due secoli dopo la morte del condottiero romano, segna l'ingresso della sostituzione monoalfabetica sistematica nella storia documentata. Il principio è di una semplicità disarmante: ogni lettera dell'alfabeto viene sostituita con la lettera che si trova a una distanza fissa - Cesare usava uno shift di tre posizioni, quindi A diventa D, B diventa E, e così via ciclicamente. Matematicamente, è un'operazione di addizione modulare: se rappresentiamo le lettere come numeri (A=0, B=1, ..., Z=22 nell'alfabeto latino ridotto), la cifratura è semplicemente $E_k(x) = (x + k) \bmod 23$, dove k è lo shift.
La struttura è quella di un gruppo ciclico - le operazioni possono essere composte, esiste un elemento identità (shift di zero, nessuna trasformazione), ogni shift ha un inverso (lo shift opposto che decifra). È matematica elementare, ma matematica comunque, e il fatto che questa struttura algebrica emergesse naturalmente da necessità pratiche di comunicazione militare dice qualcosa di profondo sull'universalità di certi pattern matematici.
Il tallone d'Achille del cifrario di Cesare è duplice. Da un lato, lo spazio delle chiavi è ridicolmente piccolo - solo ventidue shift significativi (escludendo l'identità) - rendendo la forza bruta banale: un crittoanalista può testare manualmente tutte le possibilità in meno di mezz'ora. Ma la debolezza più profonda, e più istruttiva per ciò che verrà dopo, è altra: ogni lettera del testo chiaro è sempre mappata nella stessa lettera del testo cifrato. Questa proprietà - caratteristica di tutte le sostituzioni monoalfabetiche - significa che le proprietà statistiche della lingua vengono preservate intatte.
Se la lettera E appare con frequenza 12% nel latino, la sua cifratura (qualunque essa sia) apparirà con frequenza 12% nel testo cifrato. I pattern di ripetizione, i digrammi

comuni, la lunghezza delle parole - tutto rimane riconoscibile per chi sappia guardare. Un crittoanalista che calcola le frequenze nel testo cifrato e le confronta con le frequenze note del latino può, con ragionevole certezza e rapidamente, ricostruire la mappatura. Non servono testi lunghi - con poche centinaia di caratteri l'analisi diventa affidabile. Il cifrario di Cesare è crittograficamente debole, ma la sua importanza storica è immensa: stabilisce il concetto di chiave separabile dall'algoritmo, il principio che diventerà centrale in Kerckhoffs e nella crittografia moderna.

Svetonio menziona che Augusto usava una variante leggermente diversa (shift di uno invece che tre), ma con un'eccezione curiosa: la lettera Z veniva raddoppiata invece che ciclarsi ad A. Questa irregolarità, lungi dall'aumentare la sicurezza, la diminuisce - crea un marker riconoscibile (la presenza di ZZ nel cifrato) che facilita l'identificazione del sistema. È un esempio precoce di un principio che vedremo ripetersi: la complessità arbitraria, le eccezioni non motivate matematicamente, tendono a creare vulnerabilità piuttosto che rafforzare la protezione.

Il cifrario affine - di cui il Cesare è caso speciale - generalizza l'idea a trasformazioni della forma $E(x) = ax + b \pmod n$, richiedendo che a sia coprimo con n per garantire invertibilità. Lo spazio delle chiavi si espande (per $n=23$, abbiamo $22 \times 23 = 506$ possibilità invece di 22), ma rimane una sostituzione monoalfabetica, quindi vulnerabile all'analisi di frequenza. La lezione è chiara: aumentare lo spazio delle chiavi non basta se la struttura fondamentale rimane debole. Questo insight guiderà gli sviluppi successivi verso la poliafabeticità - la necessità di cambiare la mappatura durante la cifratura stessa.

La Rivoluzione Scientifica di al-Kindī

Il salto dal IX secolo nel califfato abbaside ci porta in un mondo intellettuale radicalmente diverso. La Casa della Sapienza a Baghdad rappresentava la più grande concentrazione di conoscenza dell'epoca, dove i testi greci venivano tradotti, la matematica indiana studiata, l'astronomia raffinata, la medicina sistematizzata. In questo ambiente di sintesi e innovazione operava Abū Yūsuf Ya'qūb ibn Isḥāq al-Kindī, poliedrico genio che produsse oltre duecento opere coprendo filosofia, matematica, ottica, musica, medicina - e crittografia.

Il suo "Manoscritto sulla Decifratura dei Messaggi Criptografici", scritto attorno all'850, è straordinario non tanto per le tecniche specifiche (alcune erano probabilmente già note empiricamente) quanto per l'approccio metodologico. Al-Kindī non offre ricette o trucchi isolati; costruisce una teoria generale basata su un'osservazione empirica fondamentale: ogni lingua naturale possiede un'impronta statistica caratteristica. Le lettere non appaiono con frequenze uguali, e queste frequenze rimangono notevolmente stabili attraverso testi diversi. Questa regolarità è proprietà intrinseca della struttura linguistica - riflette la fonologia, le convenzioni ortografiche, la morfologia della lingua.

L'intuizione di al-Kindī può essere formalizzata matematicamente, e qui la connessione con il trattato tecnico diventa particolarmente ricca: la distribuzione di frequenza può essere vista come una firma probabilistica della lingua, e la sostituzione monoalfabetica, essendo una biezione, preserva esattamente questa firma - la "rimappa" ma non la distrugge. Quindi, analizzando le frequenze nel testo cifrato e confrontandole con le frequenze note della lingua sospetta, è possibile ricostruire la mappatura.
Il metodo di al-Kindī non si limita all'analisi monocarattere ma contempla livelli successivi di sofisticazione. Se l'analisi delle lettere singole è ambigua, si passa ai bigrammi - le coppie di lettere consecutive che in ogni lingua mostrano pattern caratteristici (nel latino: QU, IN, ER; nell'arabo: AL, AN, MA). Se anche questo non basta, si considerano i trigrammi, i pattern posizionali, i vincoli grammaticali. È un approccio iterativo e multimodale, che combina statistica, linguistica e ragionamento contestuale.
Ciò che rende il lavoro di al-Kindī veramente rivoluzionario, tuttavia, non è solo il contenuto tecnico ma l'epistemologia sottostante. Sta facendo scienza nel senso moderno: osservazione sistematica (contare le frequenze su corpus estesi), generalizzazione (formulare un principio universale), testabilità (le predizioni del metodo possono essere verificate), iterazione (il processo si raffina basandosi sui risultati). La crittanalisi diventa una disciplina razionale governata da principi comprensibili, non un'arte oscura basata su intuizioni mistiche o prove ed errori non sistematiche.
Il lavoro di al-Kindī rimase confinato al mondo islamico per secoli. L'Europa medievale non ebbe accesso diretto a questi testi, e quando sviluppò la propria tradizione crittografica nel Rinascimento, lo fece largamente in modo indipendente. Eppure l'influenza indiretta fu reale - attraverso i traduttori di Toledo, attraverso figure come Fibonacci che studiarono in Nord Africa, attraverso la generale osmosi culturale tra mondo islamico e cristiano. E quando il manoscritto fu finalmente tradotto e pubblicato nel mondo occidentale (1987), il riconoscimento fu immediato: al-Kindī aveva anticipato di un millennio metodi che l'Europa avrebbe "scoperto" solo nell'Ottocento.

Il Rinascimento e l'Età Moderna (1400-1900)

La Poliafabeticità come Difesa

L'Italia del Quattrocento era un laboratorio politico di straordinaria complessità. La frammentazione in stati concorrenti - il Ducato di Milano, la Repubblica di Venezia, Firenze, lo Stato Pontificio, Napoli, più una miriade di entità minori - creava un sistema di equilibri precari dove la diplomazia era arte raffinatissima e l'intelligence essenziale. Gli ambasciatori attraversavano territori ostili, i dispacci venivano intercettati sistematicamente, le alleanze mutavano con rapidità sorprendente. In

questo contesto ogni corte sviluppava i propri sistemi crittografici, e alcune figure come Giovanni Soro a Venezia raggiungevano fama (e timore) come crittoanalisti capaci di violare le cifre straniere.

Leon Battista Alberti incarnava l'ideale rinascimentale dell'uomo universale - architetto, teorico d'arte, matematico, crittografo, letterato. La sua intuizione crittografica fondamentale, presentata nel trattato "De componendis cifris" (1467), fu radicale: per difendersi dall'analisi di frequenza che minacciava tutte le sostituzioni monoalfabetiche, bisognava cambiare l'alfabeto cifrato durante la cifratura stessa. Il dispositivo che progettò - il disco cifrante, due dischi concentrici con alfabeti che potevano essere ruotati uno rispetto all'altro - era di elegante semplicità meccanica ma di profonde implicazioni concettuali.

Il principio operativo consisteva nell'iniziare la cifratura con un particolare allineamento dei dischi (stabilito in anticipo o comunicato attraverso un indicatore), cifrare alcune parole, poi ruotare il disco interno a una nuova posizione e continuare. Ogni posizione del disco implementava un cifrario di Cesare diverso - uno shift diverso. Il risultato era che la stessa lettera del testo chiaro veniva cifrata in lettere diverse a seconda di dove appariva nel messaggio. La distribuzione di frequenza del testo cifrato si appiattiva verso l'uniformità - la firma statistica della lingua veniva progressivamente cancellata.

Matematicamente, possiamo vedere il sistema di Alberti come una sequenza di cifrari di Cesare con shift variabili. Se il messaggio viene diviso in segmenti e ogni segmento cifrato con uno shift diverso, la frequenza di una lettera cifrata nel testo completo diventa una media ponderata delle sue frequenze nei vari segmenti. Con abbastanza cambi (e shift sufficientemente diversi), questa media converge verso la distribuzione uniforme. L'Indice di Coincidenza - quella misura statistica che quantifica quanto una distribuzione si discosta dall'uniformità - decresce drasticamente. Un testo monoalfabetico in latino ha IC≈0.067; con cinque cambi di alfabeto, l'IC scende verso 0.045, avvicinandosi all'IC casuale di 0.038.

Alberti non si limitò alla poliafabeticità ma propose anche una forma di supercifratura: le cifre sul suo disco esterno (1, 2, 3, 4) servivano per codificare parole intere o frasi comuni attraverso un codebook prestabilito. Questo sistema ibrido - sostituzione letterale più codificazione di fraseologia - anticipa i nomenclatori che domineranno la diplomazia europea per i successivi tre secoli. Il vantaggio è duplice: compressione (parole lunghe diventano codici brevi) e sicurezza semantica (i significati importanti sono completamente oscurati). Lo svantaggio è la necessità di distribuire e proteggere il codebook - se catturato, compromette tutte le comunicazioni passate e future.

La vera rivoluzione di Alberti, tuttavia, non fu tanto nei dettagli tecnici quanto nel principio: aveva dimostrato che l'analisi di frequenza poteva essere sconfitta, che esisteva una difesa contro l'arma più potente del crittoanalista. Questo principio sarebbe stato sviluppato, raffinato e sistematizzato nei decenni successivi, culminando nel sistema che, per una delle più persistenti ingiustizie della storia, porta il nome di qualcun altro.

Il Cifrario di Vigenère e l'"Indécifrabile"

Blaise de Vigenère, diplomatico francese del XVI secolo, pubblicò nel 1586 un trattato monumentale - "Traicté des Chiffres" - che copreva l'intera gamma della crittografia conosciuta all'epoca. Il sistema polialfabetico che porta il suo nome, tuttavia, era stato effettivamente inventato da Giovan Battista Bellaso nel 1553. Vigenère lo sistematizzò, lo rese più accessibile attraverso la famosa tabula recta (una matrice 26×26 di alfabeti traslati), ma l'innovazione cruciale - usare una parola chiave memorabile ripetuta ciclicamente invece di una progressione automatica - era di Bellaso.
Il sistema funziona così: mittente e destinatario concordano una parola chiave (diciamo "CHIAVE"). Per cifrare un messaggio, la chiave viene ripetuta sotto il testo chiaro fino a coprirne tutta la lunghezza:

Testo chiaro: A T T A C C O A L L A L B A
Chiave: C H I A V E C H I A V E C H

Ogni lettera del testo chiaro viene cifrata usando lo shift indicato dalla lettera chiave corrispondente: A+C (shift 2) dà C, T+H (shift 7) dà A, T+I (shift 8) dà B, e così via. Matematicamente, per ogni posizione i: C_i = (P_i + K_{i mod m}) mod 26, dove m è la lunghezza della chiave.
La genialità del sistema sta nella sua semplicità operativa combinata con la robustezza crittografica. La chiave può essere una parola memorabile - nessun dispositivo fisico da trasportare, nessuna tabella da nascondere - ma la sicurezza, se la chiave è sufficientemente lunga e casuale, è formidabile. Per una chiave di dieci lettere, lo spazio delle chiavi è 26^10 ≈ 1.4×10^14 - astronomicamente più grande del cifrario di Cesare (23) o anche dell'affine generale (506). La forza bruta diventa impraticabile.
Ma la vera protezione viene dall'appiattimento statistico. Con una chiave di lunghezza m, ogni lettera del testo chiaro viene potenzialmente cifrata in m lettere diverse (una per ogni posizione nella chiave). La frequenza di una lettera cifrata nel testo finale dipende non solo dalla frequenza della corrispondente lettera chiara ma anche dalla distribuzione delle lettere chiave - e se la chiave è ragionevolmente casuale, questo causa mescolamento e appiattimento delle frequenze. L'Indice di Coincidenza, quella misura che per i testi monoalfabetici rimaneva alto (~0.067 per latino/italiano), scende verso i valori casuali (~0.038). L'analisi di frequenza tradizionale fallisce.
Per due secoli il cifrario di Vigenère (correttamente implementato con chiave lunga e casuale) fu considerato inviolabile - il "chiffre indéchiffrable". Diplomatici e militari si affidavano ad esso con fiducia assoluta. Solo nell'Ottocento questa aura di invincibilità venne infranta, ma per comprendre come, dobbiamo prima considerare alcune vulnerabilità che neppure i progettisti originali avevano pienamente anticipato.

La debolezza del Vigenère non sta nella matematica fondamentale ma nelle procedure operative e nelle scelte umane. Gli operatori tendevano a usare parole chiave brevi e memorabili - spesso troppo brevi. Una chiave di cinque lettere significa che il sistema ha periodicità 5: la sesta lettera del testo chiaro è cifrata con lo stesso shift della prima, la settima con lo stesso della seconda, e così via. Questa periodicità è riconoscibile statisticamente. Inoltre, gli esseri umani sono pessimi generatori di casualità: le parole chiave tendevano a seguire pattern linguistici (vocali e consonanti alternate, sequenze comuni nella lingua), riducendo l'entropia effettiva. E infine, l'uso ripetuto delle stesse chiavi su messaggi diversi creava opportunità per confronti e analisi che un crittoanalista paziente poteva sfruttare.

La Caduta dell'"Indécifrabile": Kasiski e Friedman

Friedrich Wilhelm Kasiski, maggiore prussiano in pensione, pubblicò nel 1863 un libro modesto - "Die Geheimschriften und die Dechiffrir-Kunst" - che dimostrava come il cifrario di Vigenère potesse essere violato sistematicamente. La sua intuizione era elegante: le ripetizioni casuali nel testo chiaro producono ripetizioni visibili nel testo cifrato se e solo se sono cifrate con la stessa porzione di chiave. Se il testo chiaro contiene "THE...THE" e le due occorrenze sono distanti un multiplo della lunghezza della chiave m, allora entrambe produrranno lo stesso trigramma cifrato. Analizzando sistematicamente queste ripetizioni e calcolando le distanze tra loro, si può dedurre m (o un suo multiplo) come massimo comun divisore delle distanze.
Una volta determinata la lunghezza della chiave, il problema si riduce: il testo cifrato può essere diviso in m sottotesti, ciascuno dei quali è cifrato con uno shift costante - essenzialmente m cifrari di Cesare indipendenti. Su ogni sottotesto, l'analisi di frequenza tradizionale funziona perfettamente. Si ricostruisce lettera per lettera la chiave, e il messaggio cade.
Il metodo di Kasiski funziona bene per testi di lunghezza ragionevole (diciamo >500 caratteri) con chiavi moderate (m<20). Per testi molto lunghi con chiavi molto corte è quasi infallibile. Ma presenta limitazioni: richiede che il testo sia sufficientemente lungo da contenere ripetizioni significative, e può essere ingannato da ripetizioni casuali che producono distanze spurie. Inoltre, determinare m con precisione può essere problematico se il testo non è abbondante.
William Friedman, genio crittanalitico che operò tra le due guerre mondiali, sviluppò negli anni '20 un metodo complementare basato su un concetto matematico più profondo: l'Indice di Coincidenza. Friedman osservò che se dividiamo un testo cifrato Vigenère in sottotesti secondo la periodicità della chiave, ogni sottotesto dovrebbe avere IC vicino a quello della lingua originale (perché è monoalfabetico), mentre se dividiamo con periodicità sbagliata, i sottotesti contengono shift misti e hanno IC vicino al casuale.
Il metodo di Friedman procede così: si calcola l'IC del testo cifrato completo (dovrebbe essere intermedio tra lingua e casuale se è Vigenère); si stima la lunghezza della chiave

usando la formula m ≈ (IC_lingua - IC_casuale)/(IC_osservato - IC_casuale); si testa questa stima dividendo il testo in m sottotesti e calcolando l'IC medio - se è vicino a IC_lingua, m è corretto; se è vicino a IC_casuale, m è sbagliato. Una volta determinato m, si procede come in Kasiski con l'analisi di frequenza sui sottotesti.
I metodi di Kasiski e Friedman sono complementari: Kasiski è più intuitivo e non richiede calcoli statistici sofisticati, ma dipende dalla presenza di ripetizioni evidenti; Friedman è più robusto statisticamente e funziona anche con testi relativamente brevi o chiavi lunghe, ma richiede maggiore sofisticazione matematica. Insieme, rappresentano la morte del "chiffre indéchiffrable" - per la fine dell'Ottocento, ogni crittoanalista competente poteva violare un Vigenère standard in ore o giorni al massimo.
La lezione storica è profonda: anche sistemi che sembravano matematicamente solidi, che avevano resistito per secoli, caddero quando vennero sviluppati gli strumenti teorici appropriati. La sicurezza crittografica non è mai assoluta ma sempre relativa - dipende dalle conoscenze e risorse dell'attaccante, e queste evolvono. Il Vigenère rimaneva sicuro finché i crittoanalisti non possedevano la teoria statistica necessaria per attaccarlo; una volta sviluppata quella teoria, la sua inadeguatezza divenne evidente.

I Cifrari Avanzati e la Complessità Procedurale

Parallelamente al Vigenère, il periodo rinascimentale e moderno vide lo sviluppo di numerosi sistemi che cercavano la sicurezza attraverso vie diverse - non solo la poliafabeticità ma anche la complessità procedurale, la composizione di metodi diversi, l'uso di codebook elaborati. Il Grand Chiffre di Luigi XIV, sviluppato dai crittografi di corte Rossignol (padre e figlio) nel 1626, rappresenta un esempio paradigmatico di questo approccio.
Il sistema era un nomenclatore avanzato - un ibrido tra sostituzione e codebook. Usava 587 simboli numerici, ognuno rappresentante una sillaba francese comune, una lettera singola, una parola intera, o una frase frequente. L'omofonia - multipli numeri per la stessa sillaba - serviva ad appiattire le frequenze. Le "trappole" - numeri che modificavano l'interpretazione dei simboli precedenti o successivi - introducevano ambiguità deliberata. I "null" - numeri senza significato - confondevano il conteggio statistico. Il sistema era sufficientemente complesso che rimase inviolato per oltre due secoli, fino a quando Étienne Bazeries lo decifrò nel 1893 attraverso paziente analisi contestuale di messaggi storici archiviati.
Il cifrario di Playfair (1854), inventato da Charles Wheatstone ma promosso da Lord Lyon Playfair, introduceva un'innovazione diversa: operare su digrammi (coppie di lettere) invece che caratteri singoli. I 26 caratteri dell'alfabeto inglese venivano ridotti a 25 (combinando I/J) e disposti in una matrice 5×5 determinata da una parola chiave. Le regole di cifratura dipendevano dalla geometria: coppie sulla stessa riga venivano sostituite con le lettere immediatamente a destra; stessa colonna, con le lettere

immediatamente sotto; diagonali opposte di un rettangolo, con le altre diagonali. Il risultato era che ogni lettera poteva essere cifrata in modi diversi a seconda del suo partner, e l'analisi di frequenza monocarattere diventava inefficace.

Playfair fu usato dall'esercito britannico fino alla Prima Guerra Mondiale, testimonianza della sua robustezza pratica. Non era invulnerabile - l'analisi di frequenza sui digrammi, con testo sufficiente, poteva violare il sistema - ma offriva una buona combinazione di sicurezza e usabilità operativa. La sua importanza storica sta nell'aver dimostrato che operare su unità maggiori del singolo carattere (un principio che ritroveremo nei cifrari a blocchi moderni) offre vantaggi crittografici significativi.
Il cifrario ADFGVX, sviluppato dall'esercito tedesco nel 1918 durante la Grande Guerra, rappresenta la sintesi di principi diversi. Combinava una sostituzione (matrice 6×6 che mappava l'alfabeto più le cifre in coppie di lettere ADFGVX) con una trasposizione colonnare. La scelta delle lettere ADFGVX era motivata dalla loro massima distanza nel codice Morse - minimizzare gli errori di trasmissione radio. Il sistema era sufficientemente robusto che causò un blackout dell'intelligence alleata quando venne introdotto. Solo il genio di Georges Painvin, crittoanalista francese, riuscì a violarlo dopo mesi di lavoro intenso, sfruttando sottili irregolarità nel meccanismo di stepping dei rotori e proprietà statistiche dei testi tedeschi.
Questi sistemi - Grand Chiffre, Playfair, ADFGVX - rappresentano l'apice della crittografia manuale o semi-meccanica. Erano complessi da implementare, richiedevano operatori addestrati, e operavano al limite di ciò che poteva essere fatto con carta, matita e tabelle. Ma tutti condividevano una limitazione fondamentale: la velocità. Cifrare e decifrare messaggi lunghi richiedeva tempo, fatica, e attenzione al dettaglio. Gli errori erano comuni, e questi errori spesso diventavano vulnerabilità sfruttabili dai crittoanalisti. Il passo successivo nell'evoluzione crittografica avrebbe richiesto meccanizzazione - e quella meccanizzazione arrivò negli anni '20 del XX secolo con le macchine a rotori.

L'Era delle Macchine (1918-1945)

Enigma: L'Apice della Crittografia Elettromeccanica

Quando Arthur Scherbius presentò la sua macchina cifrante nel 1923 - chiamata Enigma - non stava semplicemente offrendo un dispositivo più veloce per fare ciò che si faceva a mano. Stava introducendo un salto qualitativo nella complessità crittografica raggiungibile. Enigma era una macchina elettromeccanica basata su rotori - dischi contenenti cablaggi elettrici che implementavano permutazioni complesse. L'idea dei rotori non era completamente nuova, ma Scherbius la perfezionò in un dispositivo pratico e commercialmente valido.

Il cuore di Enigma erano tre rotori (selezionabili da un set di cinque nella versione Wehrmacht standard) montati su un asse comune. Ogni rotore implementava una permutazione fissa ma poteva essere posizionato in una qualsiasi delle 26 orientazioni possibili. Dopo ogni lettera cifrata, il rotore destro avanzava di una posizione, cambiando la permutazione effettiva - come un contachilometri meccanico, ma con 26 "cifre" invece di 10. Quando il rotore destro completava un giro, faceva avanzare il rotore medio; quando il medio completava un giro, faceva avanzare il sinistro.
La genialità del design di Scherbius includeva il riflettore - un componente fisso che, dopo che il segnale elettrico aveva attraversato i tre rotori in una direzione, lo "rifletteva" indietro attraverso gli stessi rotori in direzione inversa. Questa simmetria aveva conseguenze eleganti: la cifratura e la decifratura usavano la stessa configurazione della macchina (premendo "A" si illuminava "Q", premendo "Q" si illuminava "A"). Per le operazioni militari, questa proprietà era preziosa - semplificava drasticamente addestramento e procedure. Ma aveva anche una conseguenza matematica inevitabile: nessuna lettera poteva mai essere cifrata in se stessa. Questa proprietà, all'apparenza innocua, sarebbe diventata il tallone d'Achille del sistema.
La versione militare aggiungeva un componente ulteriore: il plugboard (Steckerbrett), un pannello frontale con 26 jack che permetteva di connettere 10 coppie di lettere, implementando una trasposizione aggiuntiva prima e dopo i rotori. Questo moltiplicava drammaticamente lo spazio delle chiavi: la scelta di quali 3 rotori usare e in quale ordine, le posizioni iniziali, gli anelli di regolazione, e il plugboard combinavano per produrre circa 10^23 configurazioni possibili - un numero così grande da rendere la forza bruta assolutamente impensabile con la tecnologia dell'epoca.
La procedura operativa Wehrmacht prevedeva che ogni giorno venisse distribuita una configurazione di base (Grundschlüssel) attraverso codebook: quale ordine di rotori, quali posizioni degli anelli, quali connessioni del plugboard. Ogni operatore, per ogni messaggio, sceglieva poi una chiave messaggio (tre lettere casuali) e la cifrava due volte con una posizione iniziale standard, trasmettendo questo "indicatore" all'inizio del messaggio. Il destinatario, conoscendo il Grundschlüssel, decifrava l'indicatore per ottenere la chiave messaggio, impostava i suoi rotori di conseguenza, e decifrava il resto.
Questa procedura - in particolare la doppia cifratura della chiave messaggio - conteneva una vulnerabilità che i progettisti tedeschi non avevano previsto. Ma per comprenderla e sfruttarla, ci sarebbero voluti matematici di genio, determinazione incrollabile, e quello che possiamo solo chiamare il primo computer della storia moderna.

La Violazione Polacca: Matematica contro Macchina

Quando nel 1932 tre giovani matematici polacchi - Marian Rejewski, Jerzy Różycki, e Henryk Zygalski - furono reclutati dal Biuro Szyfrów (l'ufficio delle cifre polacco) per lavorare su Enigma, la Polonia si trovava nella posizione geopolitica più precaria immaginabile: schiacciata tra una Germania revanchista a ovest e un'Unione Sovietica ostile a est. L'intelligence era letteralmente questione di sopravvivenza nazionale. I metodi tradizionali della crittanalisi - linguistica, pattern matching, intuizione - si erano rivelati completamente inadeguati contro Enigma. Serviva un approccio radicalmente diverso.

L'intuizione cruciale di Rejewski fu di non attaccare Enigma come un linguista ma come un matematico. Invece di cercare di decifrare direttamente i messaggi, decise di studiare la struttura matematica delle permutazioni implementate dalla macchina. L'indicatore doppio - quella procedura dove la chiave messaggio veniva cifrata due volte - creava una relazione matematica precisa: le posizioni 1 e 4 del testo cifrato rappresentavano entrambe la stessa lettera del testo chiaro, ma con i rotori in configurazioni distanti esattamente tre step.

Questa relazione definiva una permutazione S che mappava la lettera in posizione 1 alla lettera in posizione 4. Raccogliendo dati da molti messaggi dello stesso giorno (tutti cifrati con lo stesso Grundschlüssel), Rejewski poteva ricostruire questa permutazione S quasi completamente. E qui veniva il colpo di genio: il plugboard, pur avendo uno spazio di configurazioni enorme (circa 10^14 possibilità), influenzava S in modo predicibile. Specificamente, non cambiava la struttura ciclica di S - solo "rinominava" le lettere all'interno dei cicli.

Questo significava che la struttura ciclica di S - le lunghezze dei cicli nella sua decomposizione - dipendeva solo dall'ordine dei rotori e dalle posizioni degli anelli, non dal plugboard. Con circa 105.000 configurazioni possibili di rotori+anelli (molto meno di 10^23!), Rejewski poteva catalogarle sistematicamente. Il progetto fu monumentale - richiedeva calcolare manualmente le strutture cicliche per ogni configurazione, un lavoro che impegnò un team di circa 20 persone per oltre tre anni.

Ma quando il catalogo fu completo, la procedura operativa divenne: intercettare il traffico del giorno, estrarre gli indicatori, costruire le permutazioni S, T, U (per le coppie di posizioni 1↔4, 2↔5, 3↔6), identificare le strutture cicliche, cercare nel catalogo, trovare la configurazione di rotori+anelli. Il plugboard veniva poi determinato separatamente attraverso analisi linguistica sul testo parzialmente decifrato.

Dal 1933 al 1938, il Biuro Szyfrów decifrò sistematicamente il traffico Enigma tedesco, fornendo all'intelligence polacca una finestra invalutabile sui preparativi militari tedeschi - l'espansione dell'esercito in violazione del Trattato di Versailles, i piani per l'annessione dell'Austria, la mobilitazione per la crisi dei Sudeti. Ma il 15 dicembre 1938, il blackout: la Germania introdusse due nuovi rotori (IV e V), espandendo le possibili configurazioni da 6 a 60. Il catalogo polacco divenne obsoleto istantaneamente, e non c'erano risorse per costruirne uno nuovo dieci volte più grande.

Zygalski sviluppò un metodo alternativo basato su "fogli perforati" che sfruttavano irregolarità statistiche nel meccanismo di stepping dei rotori, ma era laborioso e lento. La situazione era disperata. E poi, in un gesto di straordinaria lungimiranza strategica, il 25 luglio 1939 - cinque settimane prima dell'invasione tedesca della Polonia - il Biuro Szyfrów organizzò un incontro segreto a Pyry, vicino Varsavia, con rappresentanti britannici e francesi. E consegnò tutto: i metodi di Rejewski, il catalogo (o almeno porzioni significative), i progetti delle "bombe" (macchine elettromeccaniche per automatizzare la ricerca), i fogli di Zygalski, e persino due macchine Enigma ricostruite.
Questo regalo - forse il trasferimento di intelligence più consequenziale della storia - diede agli Alleati un vantaggio che si sarebbe rivelato decisivo. Senza Pyry, Bletchley Park non sarebbe mai esistita nella forma che conosciamo. I britannici avrebbero continuato con approcci empirici probabilmente senza successo significativo fino a troppo tardi nella guerra. La Polonia, invasa e occupata, aveva seminato i semi della vittoria alleata.

Bletchley Park e la Nascita del Computing

Quando Alan Turing arrivò a Bletchley Park nel settembre 1939, aveva già alle spalle il suo paper rivoluzionario del 1936 - "On Computable Numbers" - che aveva definito formalmente il concetto di algoritmo e computazione. A Bletchley, quel genio teorico si sarebbe cimentato con il problema pratico più pressante dell'epoca: decifrare Enigma su scala industriale, abbastanza velocemente da essere tatticamente utile.
I metodi polacchi funzionavano, ma richiedevano tempo - troppo tempo per le esigenze operative della guerra. Serviva automazione, velocità, un modo per testare migliaia di configurazioni in ore invece che settimane. Turing sviluppò un approccio basato su "cribs" - segmenti di testo chiaro ipotizzabili. I messaggi militari tedeschi contenevano formule stereotipate: "Wettervorhersage" (previsione meteo) veniva trasmesso ogni mattina alle 6:00; "Keine besonderen Ereignisse" (nessun evento particolare) era il rapporto standard quando non accadeva nulla; i nomi di luoghi, le coordinate geografiche, le sigle militari ricorrevano frequentemente.
Quando un crib veniva allineato con il testo cifrato, creava una serie di equazioni che la configurazione corretta di Enigma doveva soddisfare simultaneamente. E qui interveniva la proprietà fatale del riflettore: nessuna lettera si cifrava in se stessa. Quindi se il crib ipotizzato aveva una 'E' in una posizione dove il cifrato aveva una 'E', quella posizione veniva esclusa immediatamente - il crib era mal allineato o sbagliato. Questo drastico taglio dello spazio di ricerca, combinato con i "loop" (sequenze di connessioni che dovevano chiudersi coerentemente), rendeva possibile testare le configurazioni molto più rapidamente di una ricerca esaustiva pura.
Ma "rapidamente" è relativo. Testare configurazioni manualmente, anche con questi tagli, rimaneva proibitivamente lento. La risposta di Turing fu la "bomba" - non un'arma esplosiva ma una macchina elettromeccanica il cui nome derivava dalle

"bombe" polacche ma con design completamente diverso. La bomba di Turing (perfezionata dall'innovazione cruciale di Gordon Welchman - il "diagonal board") era essenzialmente un Enigma simulato e moltiplicato: 36 set di rotori fisici che ruotavano in parallelo, testando configurazioni a circa 2 posizioni al secondo. Con i tagli dall'analisi dei cribs e dei loop, una bomba poteva trovare la configurazione giornaliera in circa 20 minuti in media.

La produzione di bombe fu massiccia: 6 nel 1940, 210 nel 1945, operate da circa 2000 WRNS (Women's Royal Naval Service) in turni 24/7. La routine operativa divenne: intercettazione notturna del traffico tedesco, identificazione dei cribs (ore 5-6 AM), costruzione dei "menu" (diagrammi dei loop per configurare le bombe), setup e run delle bombe (ore 7-9), verifica manuale delle configurazioni candidate, break del giorno (ore 9-10), decifratura di massa di tutti i messaggi intercettati (ore 10-14), analisi intelligence e disseminazione (pomeriggio).

L'intelligence prodotta - nome in codice Ultra - era di valore inestimabile: movimenti di truppe, piani offensivi, disposizioni delle unità, forze disponibili, logistica. Influenzò ogni teatro di guerra: la Battaglia d'Inghilterra (permettendo alla RAF di allocare ottimalmente i caccia), la Battaglia dell'Atlantico (routing evasivo dei convogli per evitare i wolfpack di U-Boot), il Nord Africa (El Alamein fu vinto anche grazie alla conoscenza delle carenze di carburante di Rommel), il D-Day (conferma che l'inganno su Calais aveva funzionato). Gli storici stimano che Ultra accorciò la guerra di 2-4 anni, salvando forse 14-21 milioni di vite.

Ma la vera eredità di Bletchley va oltre l'intelligence. Le bombe erano specializzate ma rappresentavano comunque computazione automatizzata su larga scala. Il Colossus - sviluppato da Tommy Flowers nel 1943 per attaccare Lorenz, un cifrario ancora più complesso usato dal comando tedesco - era ancora più avanzato: valvole termoioniche invece di relè elettromeccanici, programmabile (seppur primitivamente), capace di operazioni logiche e conteggi a velocità senza precedenti. Era, di fatto, uno dei primi computer elettronici digitali della storia. L'architettura del Colossus influenzò i computer del dopoguerra, inclusi ENIAC e i primi progetti britannici. L'informatica moderna deve molto a Bletchley Park - nacque, letteralmente, dalla necessità di violare codici.

La lezione di Enigma è multiforme. Tecnicamente, dimostra che anche sistemi con spazio delle chiavi astronomico possono essere vulnerabili se contengono debolezze strutturali (il riflettore involutorio, gli indicatori doppi). Operativamente, sottolinea che le procedure umane sono spesso l'anello debole - gli operatori tedeschi che sceglievano chiavi messaggio non casuali (AAA, QWE, le iniziali della fidanzata), che re-inviavano messaggi con chiavi diverse fornendo "depth", che usavano formule stereotipate senza variazione. Strategicamente, conferma che l'intelligence crittografica può essere decisiva - Ultra fu un moltiplicatore di forza che permise agli Alleati di combattere più efficacemente con meno risorse. E epistemologicamente, segna il momento in cui la crittanalisi divenne computation - non più carta e matita ma macchine, algoritmi, il precursore diretto della sicurezza informatica moderna.

L'Era Digitale (1945-Oggi)

La Teoria dell'Informazione e la Crittografia Scientifica

Se Enigma e Bletchley rappresentano l'apice della crittografia elettromeccanica, l'immediato dopoguerra vide un salto concettuale ancora più profondo: la matematizzazione formale della crittografia come disciplina scientifica. Claude Shannon - brillante matematico e ingegnere del MIT che aveva lavorato durante la guerra su problemi di comunicazione sicura - completò nel 1945 un rapporto classificato che sarebbe stato declassificato e pubblicato solo nel 1949: "Communication Theory of Secrecy Systems". Questo documento, insieme al suo celebre articolo del 1948 "A Mathematical Theory of Communication", pose le fondamenta della teoria dell'informazione e della crittografia moderna.
Shannon introdusse il concetto di entropia - una misura matematica della quantità di informazione o, equivalentemente, dell'incertezza. Per una variabile aleatoria discreta X con valori possibili e relative probabilità, l'entropia $H(X) = -\Sigma P(x_i) \log_2 P(x_i)$ quantifica, in bit, il numero medio di bit necessari per codificare un messaggio dalla sorgente X. Un lancio di moneta equa ha entropia 1 bit; un dado a sei facce ha entropia $\log_2(6) \approx 2.58$ bit. La lingua inglese, considerando le correlazioni tra lettere, le parole comuni, la grammatica, ha entropia circa 1.0-1.5 bit per carattere - molto inferiore ai $\log_2(26) \approx 4.7$ bit che servirebbero se ogni lettera fosse equiprobabile e indipendente.

Questa ridondanza della lingua - circa 68% per l'inglese - è esattamente ciò che rende possibile la crittanalisi basata su pattern. Se la lingua fosse puramente casuale, non ci sarebbero frequenze caratteristiche da sfruttare, nessun digramma comune da riconoscere, nessuna struttura grammaticale da dedurre. La crittanalisi funziona perché la lingua ha struttura, e i cifrari deboli preservano tracce di quella struttura. Il compito della buona crittografia diventa quindi: eliminare o mascherare completamente quella struttura, rendere il testo cifrato statisticamente indistinguibile dal casuale.
Shannon formulò rigorosamente il concetto di perfect secrecy (segretezza perfetta): un sistema crittografico ha perfect secrecy se osservare il testo cifrato non fornisce nessuna informazione sul messaggio originale - formalmente, $P(M|C) = P(M)$ per ogni messaggio M e ogni testo cifrato C. È una proprietà di idealità assoluta. E Shannon dimostrò un teorema notevole: un sistema ha perfect secrecy se e solo se la chiave ha entropia almeno pari a quella del messaggio ($H(K) \geq H(M)$), se ogni coppia (messaggio, testo cifrato) è collegata da esattamente una chiave, e se tutte le chiavi sono equiprobabili.
Esiste un sistema - uno solo - che soddisfa questi requisiti: il one-time pad (OTP). Il principio è elementare: la chiave K è una sequenza perfettamente casuale della stessa lunghezza del messaggio M; la cifratura è semplicemente $C = M \oplus K$ (XOR bit a bit);

ogni chiave viene usata una sola volta. La perfect secrecy è immediata: per qualsiasi testo cifrato C osservato, ogni possibile messaggio M è ugualmente probabile perché esiste esattamente una chiave $K = M \oplus C$ che li collega, e tutte le chiavi sono equiprobabili ex hypothesi.
Il problema pratico dell'OTP è devastante: la chiave deve essere lunga quanto il messaggio, perfettamente casuale (il vero random è costoso), segreta (deve essere trasmessa/distribuita sicuramente), e usata una sola volta (riutilizzare la chiave distrugge la sicurezza). Se possiamo trasmettere sicuramente una chiave lunga quanto il messaggio, perché non trasmettere direttamente il messaggio? L'OTP è stato usato storicamente per comunicazioni ad altissima sicurezza dove il costo è giustificato - la linea rossa Washington-Mosca durante la Guerra Fredda, comunicazioni diplomatiche critiche - ma è impraticabile per la maggior parte delle applicazioni.
Shannon introdusse anche due principi di design fondamentali per i cifrari pratici: confusion e diffusion. La confusion rende la relazione tra chiave e testo cifrato il più complessa possibile - anche un piccolo cambiamento nella chiave dovrebbe causare un cambiamento drastico e imprevedibile nel testo cifrato. Si ottiene attraverso sostituzioni non lineari, operazioni complesse. La diffusion dissipa la struttura statistica del testo chiaro attraverso il testo cifrato - ogni bit del testo chiaro dovrebbe influenzare molti bit del testo cifrato. Si ottiene attraverso permutazioni, mixing. Nessuna singola operazione può fornire simultaneamente confusion e diffusion ottimali; la soluzione è alternarle in multipli "round". Questo principio guiderà il design di tutti i cifrari a blocchi moderni.

Il Data Encryption Standard: Crittografia Industriale

L'avvento dei computer digitali negli anni '50 e '60 trasformò radicalmente la crittografia. Non più alfabeti e lettere ma bit e byte; non più cifrari manuali o dispositivi meccanici ma algoritmi software eseguiti a velocità elettronica. Ma questa transizione creò anche un vuoto: non esisteva uno standard pubblico per la crittografia civile. Le banche necessitavano di proteggere le transazioni, le aziende volevano comunicare sicuramente, ma non c'era un algoritmo universalmente accettato, validato, efficiente.
Nel 1973 il National Bureau of Standards (predecessore del NIST) lanciò una richiesta pubblica per un algoritmo di cifratura standard. IBM rispose con una versione modificata di Lucifer, un cifrario a blocchi sviluppato dal team di Horst Feistel. Dopo valutazione e alcune modifiche suggerite dalla NSA, il Data Encryption Standard (DES) fu pubblicato nel 1977 come standard federale. Era un cifrario a blocchi: operava su blocchi di 64 bit (8 byte), usando una chiave di 56 bit effettivi (64 bit con 8 di parità), attraverso 16 round di trasformazione.
La struttura di Feistel - l'architettura sottostante del DES - è di eleganza matematica notevole. Divide il blocco di input in due metà (L_o, R_o); ogni round applica una funzione

arbitraria F (che non necessita di essere invertibile!) alla metà destra, combina il risultato con la metà sinistra via XOR, poi scambia le metà: L_{i+1} = R_i, R_{i+1} = L_i ⊕ F(R_i, K_i) dove K_i è la sottochiave del round i-esimo. La magia è che questa costruzione è automaticamente invertibile: per decifrare, si applica lo stesso procedimento ma con le sottochiavi in ordine inverso. Non serve invertire F - la struttura stessa garantisce la reversibilità.
La funzione F nel DES è dove avviene la vera crittografia. Opera su 32 bit di input e 48 bit di sottochiave attraverso una sequenza di espansione (duplicare alcuni bit), XOR con la sottochiave, sostituzione tramite S-box (8 sostituzioni non lineari che mappano 6 bit in 4 bit), e permutazione. Le S-box erano il cuore della confusion - tavole di lookup apparentemente arbitrarie ma in realtà progettate con criteri precisi per resistere agli attacchi. La permutazione forniva la diffusion. Il processo veniva ripetuto 16 volte, con sottochiavi derivate dalla chiave principale attraverso uno schedule elaborato.
Quando il DES fu proposto, scoppiò una controversia nella comunità scientifica. La chiave di 56 bit sembrava sospettosamente corta - IBM aveva proposto inizialmente 128 bit, ma la NSA aveva suggerito la riduzione. Il sospetto era che la NSA volesse poter violare il DES con i suoi supercomputer mantenendo fuori altri attori. Le S-box sembravano generate arbitrariamente senza spiegazione dei criteri di design - backdoor intenzionali? Anni dopo, quando i criteri furono declassificati, emerse che le S-box erano state fortificate contro la crittanalisi differenziale - una tecnica scoperta pubblicamente solo negli anni '90 ma già nota alla NSA negli anni '70! Il DES era solido, ma la chiave breve rimaneva la sua vera debolezza.
Nel 1998, la Electronic Frontier Foundation costruì "Deep Crack" - una macchina specializzata da $250.000 che trovava una chiave DES in media in 4.5 giorni. Oggi, con computer moderni, il DES si viola in ore. La soluzione temporanea fu il Triple DES (3DES): applicare il DES tre volte con chiavi diverse, espandendo la sicurezza effettiva a circa 112 bit. Ma 3DES era lento (tre volte le operazioni del DES) e rappresentava chiaramente una toppa piuttosto che una soluzione fondamentale.

L'Advanced Encryption Standard: Il Cifrario del XXI Secolo

Nel 1997, riconoscendo l'obsolescenza imminente del DES, il NIST lanciò un processo pubblico di selezione per un nuovo Advanced Encryption Standard (AES). I requisiti erano stringenti: blocchi di 128 bit, chiavi di 128/192/256 bit, sicurezza contro tutti gli attacchi noti, efficienza sia in software che in hardware, semplicità e analizzabilità. Quindici algoritmi furono presentati da team internazionali; dopo conferenze pubbliche, crittanalisi intensiva, e valutazioni comparative, nel 2001 Rijndael - sviluppato dai crittografi belgi Joan Daemen e Vincent Rijmen - fu selezionato vincitore e standardizzato come AES nel 2002.
Rijndael rappresenta un approccio architetturale diverso dal DES. Invece della rete di Feistel, usa una rete di sostituzione-permutazione (SP network) dove l'intero blocco

viene trasformato simultaneamente attraverso operazioni che operano direttamente sullo "state" - una matrice 4×4 di byte che rappresenta il blocco di 128 bit. Ogni round (10 per AES-128, 12 per AES-192, 14 per AES-256) applica quattro trasformazioni in sequenza: SubBytes (sostituzione non lineare dove ogni byte viene sostituito tramite una S-box basata sull'inverso moltiplicativo nel campo di Galois $GF(2^8)$), ShiftRows (permutazione che ruota ciclicamente le righe dello state), MixColumns (mixing lineare che moltiplica ogni colonna per una matrice fissa in $GF(2^8)$), e AddRoundKey (XOR con la sottochiave del round).
La bellezza matematica di AES sta nella sua coerenza algebrica. L'intera struttura è basata su operazioni nel campo finito $GF(2^8)$ - i byte sono visti come polinomi di grado 7 con coefficienti binari, e le operazioni seguono l'aritmetica del campo con polinomio riducibile $x^8 + x^4 + x^3 + x + 1$. L'S-box è costruita matematicamente (inverso moltiplicativo seguito da trasformazione affine) piuttosto che attraverso tavole apparentemente arbitrarie. MixColumns usa una matrice circolante con proprietà di massima distanza - garantisce che dopo solo due round, ogni byte di output dipenda da tutti i byte di input (diffusion rapida e completa).
Le proprietà di sicurezza sono impressionanti. La struttura soddisfa ottimamente i principi di Shannon: SubBytes fornisce confusion estrema (altissima non-linearità), ShiftRows e MixColumns forniscono diffusion completa (il "wide-trail design" assicura che dopo due round ogni bit di input influenzi uniformemente tutti i bit di output). Non esistono round-key deboli, nessuna chiave semi-debole, nessuna simmetria sfruttabile. Gli attacchi noti sono puramente teorici: per AES-128, l'attacco migliore ha complessità 2^{126} (marginalmente meglio della forza bruta 2^{128} ma completamente impraticabile); per AES-256, la complessità è 2^{254}.
Le prestazioni sono eccellenti. In software su CPU moderne senza accelerazione hardware, AES raggiunge 10-15 cicli per byte (circa 1-2 GB/s su processore single-core da 3 GHz). Con le istruzioni AES-NI (hardware dedicato presente nella maggior parte delle CPU moderne), scende a 0.6 cicli per byte (circa 20 GB/s). In hardware specializzato (FPGA/ASIC), può raggiungere throughput di 10-100 Gbps. Confrontato con i predecessori - DES lento, 3DES tre volte più lento - AES rappresenta un salto qualitativo in efficienza senza compromessi sulla sicurezza.
Oggi AES è lo standard de facto universale per la crittografia simmetrica. Lo troviamo in TLS/SSL (che protegge praticamente tutto il web), IPsec, SSH, file system cifrati (BitLocker, FileVault), database, cloud storage, applicazioni di messaggistica. È stato analizzato intensivamente per oltre vent'anni da migliaia di crittoanalisti in tutto il mondo, ha resistito a ogni attacco, e rimane sicuro. Se esiste un cifrario di cui fidarsi, è AES.

La Rivoluzione della Chiave Pubblica

Tutte le crittografie discusse finora - da Cesare a Enigma, da DES ad AES - sono simmetriche: la stessa chiave (o chiavi strettamente correlate) serve per cifrare e

decifrare. Questo crea un problema fondamentale che ha tormentato la crittografia per millenni: come possono Alice e Bob concordare una chiave segreta se non si sono mai incontrati di persona e tutti i canali di comunicazione sono insicuri? Le soluzioni tradizionali - corrieri fisici, centri di distribuzione delle chiavi, reti complesse di chiavi pre-distribuite - diventano intrattabili su larga scala. Per un milione di utenti che vogliono comunicare a coppie servirebbero circa 500 miliardi di chiavi simmetriche!
Nel 1976, due ricercatori di Stanford - Whitfield Diffie e Martin Hellman - pubblicarono "New Directions in Cryptography", un articolo che rivoluzionò completamente il campo. L'idea centrale era audace fino all'incredibile: è possibile concordare una chiave segreta su un canale pubblico completamente intercettato! Il protocollo che proposero - ora chiamato scambio di chiavi Diffie-Hellman - si basa sulla difficoltà computazionale del problema del logaritmo discreto.
Il setup pubblico consiste in un numero primo grande p (tipicamente 2048 bit) e un generatore g del gruppo moltiplicativo modulo p. Questi parametri sono pubblici, noti a tutti incluso l'attaccante Eva. Alice sceglie un numero segreto a e calcola A = g^a mod p, che invia a Bob pubblicamente. Bob sceglie un segreto b e calcola B = g^b mod p, che invia ad Alice. Alice calcola s = B^a mod p = (g^b)^a = g^(ab); Bob calcola s = A^b mod p = (g^a)^b = g^(ab). Hanno entrambi lo stesso valore s = g^(ab) mod p!
Eva vede p, g (pubblici), A = g^a mod p, e B = g^b mod p. Ma calcolare g^(ab) mod p da questi dati - senza conoscere a o b - è il problema del logaritmo discreto, computazionalmente intrattabile per p sufficientemente grande. È come se Alice e Bob avessero stabilito un segreto condiviso letteralmente di fronte agli occhi di Eva, che pur vedendo tutto non può ricostruire il segreto. La magia della matematica rende possibile l'impossibile.
Un anno dopo, nel 1977, tre ricercatori del MIT - Ronald Rivest, Adi Shamir, e Leonard Adleman - inventarono RSA, il primo sistema completo di crittografia a chiave pubblica. L'idea era ancora più radicale: ogni utente possiede una coppia di chiavi - una pubblica (distribuibile liberamente) per cifrare messaggi destinati a lui, e una privata (mantenuta segreta) per decifrare i messaggi ricevuti. La matematica sottostante è la difficoltà della fattorizzazione: dato un numero n = pq prodotto di due primi grandi, trovare p e q è computazionalmente difficile.
La generazione delle chiavi RSA procede scegliendo due primi grandi p e q (tipicamente 1024 bit ciascuno), calcolando n = p×q (il modulo RSA, 2048 bit) e ϕ(n) = (p-1)(q-1) (la funzione di Eulero), scegliendo un esponente pubblico e (comunemente e = 65537 = 2^{16} + 1), e calcolando l'esponente privato d tale che d×e ≡ 1 (mod ϕ(n)). La chiave pubblica è (n, e); la privata è (n, d). Per cifrare un messaggio M: C = M^e mod n. Per decifrare: M = C^d mod n. La correttezza deriva dal teorema di Eulero: (M^e)^d = M^(ed) = M^(1+kϕ(n)) = M × (M^ϕ(n))^k ≡ M × 1^k ≡ M (mod n).
La sicurezza si basa sulla difficoltà di fattorizzare n. Se un attaccante potesse fattorizzare n in p×q, potrebbe calcolare ϕ(n) = (p-1)(q-1) e quindi ricavare d dalla relazione d×e ≡ 1 (mod ϕ(n)). Ma il miglior algoritmo noto per fattorizzare (General Number Field Sieve) ha complessità sub-esponenziale ma comunque intrattabile per

n sufficientemente grande. RSA-2048 (chiave di 2048 bit) è considerato sicuro contro attacchi classici per il futuro prevedibile.
Un'eleganza ulteriore di RSA è che le operazioni sono commutative, permettendo firme digitali. Alice può "firmare" un messaggio M calcolando S = M^(d_A) mod n_A usando la sua chiave privata. Chiunque può verificare calcolando M' = S^(e_A) mod n_A usando la chiave pubblica di Alice - se M' = M, la firma è valida. Solo Alice, possedendo d_A, può produrre S valida. Questo fornisce autenticità, integrità, e non-ripudio - proprietà impossibili con la crittografia simmetrica pura.
L'impatto della crittografia a chiave pubblica fu rivoluzionario. Per n utenti, bastano n coppie di chiavi invece di $O(n^2)$ chiavi simmetriche. Nessun canale sicuro necessario per distribuzione delle chiavi pubbliche. Firme digitali native. Infrastruttura drasticamente semplificata. Oggi tutta la sicurezza Internet - HTTPS, PGP/GPG, SSH, certificati digitali, blockchain - si basa su crittografia a chiave pubblica. È stata, letteralmente, la tecnologia che ha reso possibile l'era digitale sicura.

Funzioni di Hash e Integrità

Accanto ai cifrari (simmetrici e asimmetrici), le funzioni di hash crittografiche costituiscono il terzo pilastro della crittografia moderna. Una funzione di hash H mappa dati di lunghezza arbitraria in una stringa di lunghezza fissa (il "digest" o "hash") - per esempio, SHA-256 mappa qualsiasi input in esattamente 256 bit. Le proprietà richieste sono tre: resistenza alla pre-immagine (dato h = H(m), impossibile trovare m'), resistenza alla seconda pre-immagine (dato m_1, impossibile trovare $m_2 \neq m_1$ con $H(m_1) = H(m_2)$), e resistenza alle collisioni (impossibile trovare qualsiasi coppia $m_1 \neq m_2$ con $H(m_1) = H(m_2)$).
Il paradosso del compleanno implica che per trovare una collisione servono solo circa 2^(n/2) tentativi per un hash di n bit - quindi SHA-256 (256 bit) richiede circa 2^{128} operazioni per trovare una collisione, computazionalmente sicuro; ma MD5 (128 bit) richiede solo 2^{64} operazioni, oggi fattibile. Di fatto, MD5 fu violato completamente: nel 2004 Wang et al. trovarono collisioni in ore; nel 2008 furono creati certificati SSL diversi con lo stesso hash MD5 - un disastro che relegò MD5 solo a checksum non critici.
SHA-1 (160 bit), sviluppato dalla NSA nel 1995, resistette più a lungo ma cadde comunque: collisioni teoriche nel 2005, e nel 2017 Google produsse la prima collisione pratica (progetto SHAttered) con costo equivalente a 110 anni-GPU. SHA-1 è ora deprecato. Le funzioni sicure oggi sono SHA-2 (famiglia che include SHA-256 e SHA-512, sviluppata dalla NSA nel 2001) e SHA-3 (Keccak, vincitore di una competizione NIST e standardizzato nel 2015, con struttura radicalmente diversa basata su "sponge construction" invece di Merkle-Damgård).
Le applicazioni delle funzioni di hash sono molteplici e fondamentali. L'integrità dei dati: confrontare l'hash di un file scaricato con l'hash pubblicato ufficialmente garantisce che il file non sia stato modificato. Le firme digitali: invece di firmare un

messaggio lungo M, si firma il suo hash H(M) - velocità (l'hash è fisso e piccolo) e sicurezza (l'hash cattura completamente il contenuto di M). La memorizzazione delle password: mai memorizzare password in chiaro! Si memorizza H(password), e anche se il database viene rubato l'attaccante non può recuperare le password (resistenza alla pre-immagine). Con un salt casuale - hash(password || salt) dove il salt è memorizzato accanto all'hash - si prevengono attacchi con rainbow tables.
Le proof-of-work nella blockchain: Bitcoin richiede di trovare un nonce tale che SHA-256(SHA-256(header_blocco || nonce)) < target, dove target è un numero molto piccolo (molti zeri iniziali). Trovare questo nonce richiede provare trilioni di valori - è la "prova di lavoro" che rende costosa la creazione di blocchi e quindi sicura la blockchain. Le HMAC (Hash-based Message Authentication Codes) combinano hashing con una chiave segreta per autenticare messaggi, fornendo sia integrità che autenticazione in un solo costrutto.

Il Futuro della Crittografia

La Minaccia Quantistica

L'intera infrastruttura crittografica moderna - RSA, Diffie-Hellman, curve ellittiche - si basa su problemi matematici considerati difficili: fattorizzazione di interi grandi, logaritmo discreto, logaritmo discreto su curve ellittiche. Difficili, però, per computer classici. Nel 1994, Peter Shor scoprì algoritmi quantistici che risolvono questi problemi in tempo polinomiale. Un computer quantistico sufficientemente grande potrebbe fattorizzare RSA-2048 in ore, risolvere DLP in minuti, rompere ECDSA istantaneamente. L'intera crittografia a chiave pubblica diventerebbe obsoleta.
I computer quantistici sfruttano sovrapposizione e entanglement quantistico per esplorare spazi di ricerca esponenzialmente grandi in parallelo. Attualmente esistono ancorché primitivi - poche decine di qubit, alto tasso di errore, temperature criogeniche, costi proibitivi. Ma il progresso è rapido: da 53 qubit (Google, 2019) a 1121 qubit (IBM, 2023). Le proiezioni suggeriscono che computer quantistici pratici per la crittanalisi potrebbero esistere tra 10-20 anni - un orizzonte preoccupantemente vicino.
L'algoritmo di Grover rappresenta una minaccia più limitata ma reale per la crittografia simmetrica. Per ricerca in database non ordinato di N elementi: classico richiede O(N), quantistico O(√N). Applicato alla forza bruta su chiavi: per chiave di n bit, classico richiede 2^n tentativi, quantistico 2^(n/2). Quindi AES-128 (128 bit di sicurezza classica) diventa effettivamente 64 bit di sicurezza quantistica - insufficiente. AES-192 diventa 96 bit - marginale. AES-256 diventa 128 bit - sicuro. La soluzione è semplice: raddoppiare le lunghezze delle chiavi simmetriche. Passare da AES-128 ad AES-256, da SHA-256 a SHA-512.
Ma per la crittografia asimmetrica non esiste soluzione semplice - non possiamo semplicemente "raddoppiare" RSA perché l'algoritmo di Shor lo rompe

indipendentemente dalla dimensione della chiave. Serve una rivoluzione: algoritmi basati su problemi matematici diversi, problemi che rimangono difficili anche per computer quantistici. Questa è la crittografia post-quantistica.

La Crittografia Post-Quantistica

Nel 2016 il NIST lanciò un processo di standardizzazione per algoritmi resistenti ai computer quantistici. I requisiti: sicurezza contro attacchi classici e quantistici, efficienza pratica, dimensioni ragionevoli di chiavi/firme/cifrati, basati su problemi matematici diversi da fattorizzazione e logaritmo discreto. Sessantanove algoritmi furono sottoposti; dopo round di valutazione pubblica e crittanalisi intensiva, nel 2022 il NIST annunciò i vincitori.
Le categorie di approcci riflettono diversi problemi matematici "hard" anche per computer quantistici. I reticoli (lattice-based) si basano sul Shortest Vector Problem e Learning With Errors - dato un sistema di equazioni lineari con rumore, trovare la soluzione è difficile. CRYSTALS-Kyber (per scambio di chiavi) e CRYSTALS-Dilithium (per firme digitali) furono selezionati come standard principali. Offrono buon equilibrio tra sicurezza, prestazioni e dimensioni (chiavi pubbliche ~1 KB, firme ~2-3 KB). La matematica sottostante - algebra lineare su reticoli - è relativamente ben compresa, con decenni di analisi alle spalle.
I codici (code-based) si basano sul Syndrome Decoding Problem - decodificare un codice lineare casuale è NP-completo. Classic McEliece, basato sui codici di Goppa proposti già nel 1978, rappresenta l'approccio più maturo (45+ anni di scrutinio). Il vantaggio è sicurezza estremamente ben fondata; lo svantaggio drammatico sono le chiavi pubbliche enormi (200 KB - 1 MB), impraticabili per molte applicazioni.
Gli hash (hash-based) costruiscono firme digitali usando solo funzioni di hash - SPHINCS+ usa alberi di Merkle e offre sicurezza basata esclusivamente sulla robustezza delle funzioni hash (minimo set di assunzioni). Chiavi pubbliche piccole (~32-64 byte) ma firme grandi (~8-50 KB) e generazione lenta. Gli isogeni (isogeny-based) sembravano promettenti con chiavi molto compatte, ma SIKE - candidato NIST - fu violato drammaticamente nel 2022 da Castryck e Decru in poche ore su PC, dimostrando quanto sia difficile progettare crittografia post-quantistica sicura.
La timeline della transizione è critica. Gli standard finali (FIPS 203, 204, 205) furono pubblicati nel 2024. Il periodo 2025-2030 vedrà implementazione nei protocolli (TLS 1.3, ecc.), aggiornamento di software/firmware, certificati ibridi che combinano algoritmi classici e post-quantistici per massima protezione durante la transizione. Entro 2030-2035 si prevede deprecazione completa degli algoritmi classici vulnerabili ai computer quantistici e migrazione totale al post-quantistico.
Ma c'è urgenza anche ora per il problema "harvest now, decrypt later": attaccanti sofisticati (stati-nazione) possono registrare oggi tutto il traffico cifrato, conservarlo, e attendere che i computer quantistici diventino disponibili per decifrarlo retroattivamente. Informazioni sensibili a lungo termine - dati medici, segreti

governativi, proprietà intellettuale - sono già a rischio. La transizione deve avvenire ora, non quando i computer quantistici saranno realtà.

Frontiere Emergenti

Oltre la minaccia quantistica, la crittografia moderna esplora frontiere che avrebbero stupito anche i visionari di cinquant'anni fa. La crittografia omomorfica permette di eseguire calcoli su dati cifrati senza doverli decifrare - cloud computing sicuro dove il server elabora dati senza mai vederne il contenuto. Le implementazioni moderne (BGV, BFV, CKKS) sono ancora lente (1000-10000× rispetto a calcoli normali) ma le applicazioni emergenti - machine learning su dati sensibili medici/finanziari, votazione elettronica, ricerca su database cifrati - giustificano l'overhead.
Le zero-knowledge proofs (ZKP) permettono di provare di conoscere un segreto senza rivelarlo. I ZK-SNARKs (succinct non-interactive arguments of knowledge) producono prove molto compatte e verificabili velocemente senza interazione. Zcash usa ZK-SNARKs per transazioni private dove mittente, destinatario e importo rimangono completamente nascosti, ma la validità della transazione è provata matematicamente. Gli zkRollups su Ethereum usano ZKP per scalabilità - elaborare migliaia di transazioni off-chain, produrre una prova singola di validità, pubblicarla on-chain.
La privacy differenziale fornisce garanzie formali sulla protezione della privacy quando si pubblicano statistiche su dataset. Apple, Google, il Census Bureau USA usano privacy differenziale per raccogliere dati aggregati proteggendo i singoli individui. Il quantum key distribution (QKD) usa le leggi della fisica quantistica per distribuire chiavi - il protocollo BB84 garantisce che qualsiasi intercettazione sia rilevabile grazie al principio di indeterminazione di Heisenberg. Implementazioni commerciali esistono, satelliti per QKD a lunga distanza sono operativi (Micius della Cina, 2016).
La crittografia leggera (lightweight cryptography) ottimizza per dispositivi IoT con risorse limitate. Lo standard NIST Ascon (2023) è progettato per microcontrollori con pochi kilobyte di RAM, consumo energetico minimo, implementazione compatta. Con miliardi di dispositivi IoT - sensori, smart home, veicoli connessi - la sicurezza efficiente su hardware limitato diventa cruciale.
L'intelligenza artificiale sta trasformando anche la crittografia. AI per crittanalisi automatica cerca vulnerabilità in algoritmi o implementazioni. Generazione automatica di prove di sicurezza verifica formalmente proprietà crittografiche. Machine learning su dati cifrati (combinando FHE con ML) permette modelli AI che operano su dati sensibili senza mai vederli in chiaro - medicina personalizzata, finanza, applicazioni dove privacy è essenziale ma analisi sofisticata necessaria.

Questo viaggio attraverso quattromila anni di crittografia rivela pattern ricorrenti e lezioni durature. La prima è che la sicurezza è sempre relativa, mai assoluta. Il "chiffre indéchiffrable" di Vigenère cadde quando furono sviluppati gli strumenti statistici

appropriati. Enigma, con il suo spazio delle chiavi astronomico, fu violata attraverso debolezze strutturali e procedurali. Il DES, standard per vent'anni, divenne obsoleto quando la potenza computazionale crebbe abbastanza. La crittografia attuale, per quanto robusta, dovrà confrontarsi con i computer quantistici. La sicurezza dipende dalle conoscenze e risorse dell'attaccante, e queste evolvono implacabilmente.

La seconda lezione è che la complessità non equivale a sicurezza. L'ADFGVX, con la sua combinazione elaborata di sostituzione e trasposizione, sembrava formidabile ma conteneva regolarità sfruttabili. Enigma era complessa ma il riflettore introduceva simmetria letale. I sistemi più sicuri - AES, RSA moderno, SHA-256 - sono relativamente semplici nei principi fondamentali, ma quella semplicità è costruita su fondamenta matematiche solide, analizzate per decenni. La sicurezza vera viene da design basato su hard problems matematici ben compresi, non da oscurità artificiale.

La terza lezione è l'importanza cruciale delle procedure operative. Enigma era matematicamente forte, ma operatori che sceglievano chiavi non casuali, messaggi con formule stereotipate, re-invii con chiavi diverse - tutti errori umani - fornirono le vie di accesso ai crittoanalisti. La migliore crittografia può essere vanificata da implementazione sciatta, gestione negligente delle chiavi, o utenti che prendono scorciatoie. La sicurezza è una catena; l'anello più debole è quasi sempre umano.

La quarta lezione è che la crittografia e la crittanalisi evolvono dialetticamente. Ogni nuovo metodo di cifratura stimola lo sviluppo di tecniche per violarlo; ogni successo crittoanalitisco spinge verso sistemi più robusti. Al-Kindī sviluppò l'analisi di frequenza; Alberti rispose con la poliafabeticità. Kasiski e Friedman violarono Vigenère; la risposta furono macchine a rotori. Rejewski e Turing sconfissero Enigma; emersero computer digitali e cifrari come DES. Questa corsa agli armamenti intellettuali ha attraversato i millenni e continua oggi con la minaccia quantistica e la risposta post-quantistica. Non esiste "vittoria finale" per nessuna delle due parti - solo evoluzione continua.

La quinta lezione è l'universalità della matematica. I principi scoperti dai babilonesi (trasformazioni reversibili complesse), formalizzati da al-Kindī (analisi statistica), raffinati da Shannon (entropia, confusion, diffusion) rimangono validi. Le strutture algebriche - gruppi, campi, permutazioni - emergono naturalmente dalla crittografia pratica millenni prima che la teoria astratta fosse sviluppata. La matematica non è invenzione arbitraria ma scoperta di pattern intrinseci alla realtà. Che questi pattern possano anche proteggere informazioni è testimonianza della profondità e versatilità del pensiero matematico.

La sesta lezione riguarda l'importanza della trasparenza e dell'analisi pubblica. I sistemi che hanno resistito meglio alla prova del tempo - DES (nonostante la chiave corta), AES, RSA, curve ellittiche - sono quelli che furono sottoposti a scrutinio pubblico intensivo. Migliaia di crittoanalisti nel mondo hanno cercato vulnerabilità; quando non ne trovano dopo decenni, la fiducia è giustificata. Al contrario, sistemi proprietari "security through obscurity" falliscono regolarmente quando finalmente vengono analizzati. Il principio di Kerckhoffs - la sicurezza deve risiedere nella chiave,

non nel metodo - si conferma ancora e ancora. L'apertura, paradossalmente, produce sicurezza migliore del segreto.
La settima e forse più importante lezione è che la crittografia non è solo disciplina tecnica ma anche profondamente umana. Nasce da necessità concrete - proteggere segreti commerciali in Mesopotamia, coordinare campagne militari per Sparta, condurre diplomazia nel Rinascimento frammentato, vincere guerre mondiali nel XX secolo. Ogni sistema crittografico porta l'impronta del suo tempo - le limitazioni tecnologiche, le strutture sociali, le priorità culturali, i conflitti geopolitici. Studiare la crittografia è anche studiare la storia umana attraverso una lente particolare: quella del segreto, della fiducia, del potere che deriva dal controllo dell'informazione.
Oggi viviamo in un'era dove la crittografia è onnipresente e invisibile. Ogni volta che visitiamo un sito HTTPS, inviamo un messaggio su Signal, effettuiamo un pagamento digitale, accediamo al nostro banking online, stiamo usando crittografia - AES, RSA, curve ellittiche, SHA-256, e molti altri strumenti che sono discendenti diretti dei sistemi descritti in questo saggio. L'infrastruttura digitale moderna letteralmente non potrebbe esistere senza crittografia forte. Il commercio elettronico, le comunicazioni private, la democrazia digitale, la protezione dei dati personali - tutto dipende da matematica che affonda radici in quattromila anni di evoluzione.
E il futuro? I computer quantistici rappresentano una minaccia esistenziale per molto della crittografia attuale, ma la risposta - la crittografia post-quantistica - è già in sviluppo, basata su nuovi hard problems che resistono agli attacchi quantistici. La crittografia omomorfica, le zero-knowledge proofs, la privacy differenziale aprono possibilità che sembravano fantascienza anche vent'anni fa - calcolare su dati cifrati, provare verità senza rivelare segreti, pubblicare statistiche accurate proteggendo la privacy individuale.
Ma qualunque forma prenda la crittografia futura, alcuni principi rimarranno: la separazione tra algoritmo e chiave, la necessità di fondamenta matematiche solide, l'importanza dell'analisi pubblica, la tensione dialettica con la crittanalisi, e l'intersezione profonda tra necessità pratiche e bellezza matematica. La crittografia è stata, è, e continuerà a essere una delle discipline più ricche e conseguenziali che l'umanità abbia sviluppato - ponte tra matematica pura e applicazione urgente, tra teoria astratta e sopravvivenza concreta, tra il desiderio di comunicare e la necessità di proteggere.

APPROFONDIMENTO

PARTE I: L'ALBA DELLA CRITTOGRAFIA (3000 a.C. - 400 d.C.)

CAPITOLO 1

I Segreti della Mesopotamia: La Nascita dell'Oscuramento Informativo

Nelle pianure alluvionali tra il Tigri e l'Eufrate, dove la civiltà umana mosse i suoi primi passi verso l'organizzazione complessa, nacque anche il primo impulso documentato verso la protezione dell'informazione. Non si trattava ancora di crittografia nel senso moderno – non esistevano le trasformazioni algoritmiche definite né le chiavi segrete – ma piuttosto di una forma sofisticata di oscuramento informativo che sfruttava la complessità intrinseca del sistema di scrittura cuneiforme per rendere certi testi inaccessibili a chi non possedesse le conoscenze specialistiche.

1.1 Il Contesto delle Tavolette Cuneiformi

Le testimonianze più antiche di un'alterazione intenzionale della scrittura provengono dai siti archeologici del II millennio a.C., in particolare dalla tavoletta CBS 1766, conservata presso il Museum dell'Università della Pennsylvania e datata con precisione al 1400±50 a.C. Questa tavoletta d'argilla, come altre del suo genere, contiene le formule per la produzione degli smalti vetrosi – una conoscenza preziosa che conferiva vantaggi economici significativi a chi la possedeva.
Il sistema cuneiforme sumero-accadico, lungi dall'essere una semplice corrispondenza uno-a-uno tra i segni e i suoni, costituiva un complesso sistema semiotico stratificato. Ogni segno poteva funzionare simultaneamente come:
- Logogramma: rappresentando un'intera parola o un concetto
- Fonogramma: indicando i valori sillabici (monoconsonantico, biconsonantico, o triconsonantico)
- Determinativo: fornendo la classificazione semantica senza alcun valore fonetico
Questa polifonia – la capacità di ogni segno di possedere multipli valori di lettura – variava non solo sincronicamente tra i diversi contesti, ma anche diacronicamente attraverso i secoli e diatopicamente tra le diverse regioni del mondo mesopotamico. Tale complessità intrinseca forniva un terreno fertile per l'oscuramento intenzionale.

1.2 Le Tecniche di Oscuramento Documentate

L'analisi paleografica delle tavolette contenenti le formule artigianali rivela dei pattern sistematici di modificazione che vanno ben oltre la variazione scribale casuale:
La sostituzione con le varianti arcaiche: Gli scribi sostituivano le forme grafiche standard con le versioni obsolete, comprensibili solo a chi possedesse una conoscenza storica approfondita del sistema di scrittura. Ad esempio, la forma paleo-babilonese del segno A ("acqua") veniva utilizzata al posto della forma neo-babilonese corrente, creando un ostacolo immediato per i lettori contemporanei abituati solo alle forme moderne.
La decomposizione logogrammatica: I logogrammi standard venivano sostituiti con le sequenze sillabiche equivalenti foneticamente ma visivamente irriconoscibili. Il logogramma URUDU ("rame") poteva essere scritto come u-ru-du, una sequenza che

richiedeva una ricostruzione fonetica attiva anziché il riconoscimento immediato del concetto.
L'omissione strategica dei determinativi: I classificatori semantici, normalmente inclusi per disambiguare le letture multiple, venivano deliberatamente omessi, aumentando massimamente l'ambiguità interpretativa e costringendo il lettore a fare affidamento su una conoscenza contestuale estensiva.
L'inversione sequenziale: L'ordine standard segno-determinativo veniva invertito, creando delle sequenze graficamente valide ma semanticamente opache per chi non fosse iniziato alle convenzioni particolari del testo.

1.3 Una Prospettiva Matematica: Perché Non è Crittografia

È cruciale comprendere perché questi sistemi, per quanto sofisticati, non costituiscono la vera crittografia secondo i criteri moderni. La crittografia richiede:

1. Una trasformazione algoritmica definita: Deve esistere una funzione $T: M \rightarrow C$ che mappa deterministicamente lo spazio dei messaggi (M) nello spazio dei testi cifrati (C)

2. La reversibilità controllata: Deve esistere una funzione inversa $T^{-1}: C \rightarrow M$ la cui applicazione richieda la conoscenza di una chiave segreta k

3. La separazione tra l'algoritmo e la chiave: La sicurezza deve risiedere nella chiave, non nell'oscurità del metodo (il principio di Kerckhoffs)

I sistemi mesopotamici violano tutti e tre questi requisiti. Non esiste una trasformazione algoritmica specificabile; la "reversibilità" dipende semplicemente dal possesso di una conoscenza specialistica distribuita socialmente; non esiste il concetto di una chiave separabile dal metodo stesso.
La protezione offerta è dunque sociologica piuttosto che matematica: funziona perché l'alfabetizzazione cuneiforme è limitata a una piccola élite scribale, e la conoscenza delle varianti arcaiche è ancora più ristretta. Qualsiasi scriba sufficientemente istruito, indipendentemente dalla sua relazione con l'autore originale, può in principio decifrare il testo. Non c'è alcun segreto condiviso, non c'è alcuna chiave.
Ciononostante, questi sistemi rappresentano il primo riconoscimento esplicito del valore strategico dell'oscuramento informativo – un'intuizione che guiderà lo sviluppo dei veri sistemi crittografici nei millenni successivi.

1.4 La Codifica Numerica Astronomica Babilonese

Parallelamente ai sistemi di oscuramento testuale, gli astronomi babilonesi del periodo neo-babilonese (626-539 a.C.) svilupparono delle tecniche sofisticate per

proteggere i coefficienti critici nei loro calcoli celesti. Qui ci avviciniamo maggiormente a una vera crittografia, poiché emerge il concetto di una trasformazione intenzionale dei dati.
Il sistema numerale sessagesimale babilonese – a base 60 – permetteva delle rappresentazioni estremamente precise delle quantità astronomiche. Un numero N veniva espresso come:

$N = \Sigma_{i= -m}^{n} a_i \times 60^i$

dove $a_i \in \{0, 1, 2, ..., 59\}$ sono le cifre sessagesimali, n rappresenta l'ordine più alto (la parte intera), e m l'estensione frazionaria. La notazione moderna usa il punto e virgola per separare gli ordini: 29;31,50 significa 29 + 31/60 + 50/3600.
Le tavolette astronomiche mostrano dei coefficienti espressi non direttamente, ma come i risultati di operazioni aritmetiche che richiedono una conoscenza contestuale per essere risolte correttamente. Ad esempio, il periodo sinodico lunare (29,530589 giorni), fondamentale per i calendari e le predizioni delle eclissi, appare sulla tavoletta ACT 18 codificato come:

$P\_codificato = [(15 + 16) \times 2] / [(2^2)^{0.5}]^2$; [31, 50, 8, 20]

La parte intera dovrebbe essere 29, ma l'espressione fornita produce 15,5 – un errore intenzionale che richiede una correzione basata sulla conoscenza astronomica indipendente. Solo chi già conosce il valore approssimativo del periodo lunare può riconoscere e correggere l'anomalia.
Questa tecnica introduce un concetto profondamente importante: la protezione attraverso una trasformazione reversibile ma computazionalmente complessa. Per un lettore non specializzato, invertire la trasformazione richiede una ricerca esaustiva nello spazio delle operazioni possibili – una complessità che cresce fattorialmente. Per chi possiede la "chiave" (la conoscenza astronomica contestuale), la decodifica è immediata.
Siamo ancora lontani dalla crittografia moderna, ma i semi concettuali sono piantati.

CAPITOLO 2

I Geroglifici e la Steganografia: L'Antico Egitto

Se la Mesopotamia ci offre i primi esempi di un oscuramento sistematico, l'Antico Egitto ci introduce a una dimensione diversa della protezione informativa: l'uso delle modificazioni grafiche non tanto per rendere il messaggio illeggibile a tutti tranne agli iniziati, ma per conferire la sacralità, il prestigio e il potere apotropaico al testo stesso.

2.1 La Struttura Tripartita del Sistema Geroglifico

Il sistema di scrittura geroglifico egiziano, in uso dal periodo Protodinastico (circa 3200 a.C.) fino al periodo tardo romano (il IV secolo d.C.), rivaleggiava con il cuneiforme in complessità stratificata. L'inventario standard del Medio Regno (2055-1650 a.C.) comprendeva circa 750 segni di uso comune, ma l'inventario completo, includendo le varianti regionali, gli arcaismi e gli hapax (i segni usati una sola volta), raggiungeva le 5000 unità.
Come il cuneiforme, il sistema geroglifico operava su tre livelli simultanei:

Gli ideogrammi rappresentavano i concetti direttamente – il sole per "giorno" o "Ra", il papiro per "verde" o "vegetazione"

I fonogrammi indicavano i valori consonantici (il vocalismo, cruciale per la pronuncia effettiva, era largamente non scritto e deve essere ricostruito), articolati in:
- Monoconsonantico (un singolo suono)
- Biconsonantico (due consonanti)
- Triconsonantico (tre consonanti)

I determinativi chiudevano le parole fornendo il contesto semantico senza contribuire alla lettura fonetica – il determinativo di "uomo" dopo un nome maschile, di "astratto" dopo i concetti filosofici, di "nemico" dopo le popolazioni ostili.

Questa ricchezza semiotica permetteva un'espressione di straordinaria precisione, ma anche un'ambiguità controllata quando desiderata.

2.2 Le Modificazioni Grafiche della XII Dinastia

Il caso più studiato di un'alterazione geroglifica sistematica proviene dalla tomba di Menet-Khufu a Beni Hassan (27°56'N, 30°53'E), datata al regno di Amenemhat II (1929-1895 a.C.) della XII Dinastia. Percy Newberry, lo scopritore della tomba nel 1893, notò immediatamente che l'iscrizione biografica principale presentava delle anomalie grafiche significative.
L'analisi paleografica quantitativa rivela 24 sostituzioni grafiche non standard su un corpus di 342 segni totali – una frequenza del 7,02%, ben oltre la variazione casuale attesa. Le sostituzioni seguono dei pattern riconoscibili:

La moltiplicazione grafica: Il segno kt (il pane, X1 nella classificazione Gardiner) appare raddoppiato (ktkt) senza alcuna motivazione grammaticale

L'arcaismo prestigioso: Il gufo ┤ (M17) è reso nella forma caratteristica dell'Antico Regno anziché nella forma contemporanea della XII Dinastia

L'elaborazione decorativa: Il pilastro djed 𓊽 (R11), il simbolo di stabilità associato a Osiride, è stilizzato in una maniera più complessa del normale

La ripetizione enfatica: La canna 𓇋 (M17), rappresentante la vocale /i/, appare triplicata in contesti dove una singola occorrenza sarebbe standard

2.3 L'Interpretazione Funzionale: Il Prestigio, Non la Segretezza

L'interpretazione moderna, consolidata attraverso il lavoro di Sir Alan Gardiner (1927), James P. Allen (2000), e Friedrich Junge (2005), converge su una comprensione non crittografica di queste modificazioni. Le funzioni identificate sono:

Il prestigio culturale: L'uso delle forme arcaiche dimostrava l'erudizione paleografica, collegando il defunto a una tradizione scritta prestigiosa

L'arcaismo reverenziale: L'Antico Regno era visto come l'età dell'oro; imitarne la scrittura conferiva la sacralità

La protezione magico-religiosa: Confondere le entità negative nell'aldilà che potrebbero tentare di interpretare malevolmente il testo

La differenziazione sociale: Marcare i testi élitari come distinti dalla scrittura ordinaria, accessibili solo ai lettori altamente istruiti

Possiamo quantificare l'efficacia dell'oscuramento. Sia L il livello di alfabetizzazione geroglifica di un lettore, definito come:

L = N_conosciuti / N_totale

dove N_conosciuti è il repertorio dei segni padroneggiati e N_totale l'inventario completo (circa 5000).

Per uno scriba medio: L_medio ≈ 750/5000 = 0,15
Per uno scriba di élite: L_élite ≈ 2000/5000 = 0,40

La probabilità che un lettore con il livello L comprenda un testo con n sostituzioni non standard che richiedono la conoscenza delle varianti rare è approssimativamente:

P(comprensione) ≈ L^n

Per il testo di Menet-Khufu con n = 24 sostituzioni:

- Scriba medio: $P \approx 0{,}15^{24} \approx 1{,}78 \times 10^{-20}$ (essenzialmente zero)
- Scriba élite: $P \approx 0{,}40^{24} \approx 4{,}72 \times 10^{-10}$ (ancora trascurabile)

Il testo funziona dunque come una barriera sociale ed educativa estremamente efficace, pur non essendo una vera crittografia. Non c'è alcuna chiave segreta, solo la conoscenza distribuita inegualmente nella società.

2.4 La Gematria Egizia: Verso la Crittografia Numerica

Nel periodo tardo (dal VII al III secolo a.C.), sotto l'influenza greca crescente, emerge in Egitto un sistema che si avvicina maggiormente alla crittografia: la gematria copta, documentata nei Papiri Magici Greci (PGM). Qui, ogni lettera dell'alfabeto copto (derivato dal greco con le aggiunte demotiche) riceve un valore numerico fisso:

ⲁ (Alef) = 1, ⲃ (Bet) = 2, ⲅ (Gimel) = 3, ... fino a ⲧ (Taw) = 400

Il valore gematrico di una parola $W = a_1a_2...a_n$ è semplicemente:

$\Gamma(W) = \Sigma_{i=1}^{n} \Phi(a_i)$
dove Φ è la funzione di mappatura lettera→numero.

Questo costituisce un vero sistema di sostituzione numerica reversibile. Il nome divino "Thoth" (ⲑⲱⲑ in copto) ha il valore:

$\Gamma(\text{ⲑⲱⲑ}) = \Phi(\text{Taw}) + \Phi(\text{Het}) + \Phi(\text{Waw}) + \Phi(\text{Taw}) = 400 + 8 + 6 + 400 = 814$

La cifratura diventa: $E(m) = \Gamma(m)$, mappando le parole nei numeri.
Tuttavia, emerge immediatamente una debolezza critica: la non unicità. La funzione Γ è surgettiva ma non iniettiva. Dato un numero c, esistono generalmente multiple parole W tali che $\Gamma(W) = c$. Per c = 814, oltre a "Thoth", esistono almeno 47 altre combinazioni di lettere che sommano allo stesso valore.
Il numero delle preimmagini $\Gamma^{-1}(c)$ cresce approssimativamente come O(c^(n-1)/n!) per le parole di lunghezza n, rendendo la decifratura ambigua senza il contesto aggiuntivo. La gematria funziona dunque meglio per i nomi propri brevi (dove il contesto risolve l'ambiguità) che per i messaggi lunghi.
Ciononostante, segna un passo concettuale importante: l'idea di trasformare il linguaggio in numeri attraverso una funzione definita, con l'intento esplicito di nascondere il significato a chi non conosce la mappatura – il precursore diretto dei cifrari di sostituzione che fioriranno nel mondo greco-romano.

CAPITOLO 3

Lo Scitale Spartano: I Primi Sistemi di Trasposizione

Lasciamo ora il Vicino Oriente antico per il mondo greco classico, dove la crittografia compie un salto qualitativo decisivo. Con lo scitale spartano incontriamo il primo dispositivo crittografico documentato che implementa una trasformazione puramente algoritmica, basata sulla trasposizione posizionale piuttosto che sull'oscuramento simbolico o sulla sostituzione.

3.1 Sparta e la Necessità Strategica

La Sparta del V secolo a.C. operava sotto dei vincoli militari unici. La sua struttura sociale rigidamente stratificata – con una minuscola élite spartiate governante su una vasta popolazione di iloti soggiogati – richiedeva una proiezione di potere costante ben oltre i confini del Peloponneso. Gli strategoi (i generali) conducevano le campagne a centinaia di chilometri da casa, attraverso i territori ostili o politicamente instabili.

Le comunicazioni dovevano attraversare:

- I territori controllati dalle poleis rivali
- Le regioni popolate dalle popolazioni di fedeltà dubbia
- Il mare aperto soggetto alla pirateria

Un messaggero intercettato con i dispacci in chiaro avrebbe potuto compromettere intere operazioni. Serviva un sistema che proteggesse i contenuti pur rimanendo praticabile nelle condizioni operative difficili del mondo antico.

3.2 La Descrizione Tecnica del Dispositivo

Lo scitale (σκυτάλη) era di un'elegante semplicità meccanica: due bastoni cilindrici di legno (tipicamente di olivo o di quercia), identici nel diametro, posseduti uno dal mittente e uno dal destinatario. Per cifrare un messaggio:

1. Una striscia sottile di cuoio o di papiro veniva avvolta strettamente intorno al cilindro del mittente
2. Il messaggio veniva scritto longitudinalmente lungo il cilindro
3. La striscia veniva rimossa: i caratteri ora appaiono sparsi lungo la striscia in un ordine apparentemente casuale
4. La striscia veniva inviata al destinatario
5. Il destinatario la avvolgeva intorno al proprio cilindro identico, rivelando il messaggio

La sicurezza risiedeva nel diametro esatto del cilindro – un parametro fisico che costituiva la "chiave" del sistema.

3.3 La Formalizzazione Matematica

Possiamo modellare lo scitale come una permutazione di trasposizione. Sia il messaggio m = $m_1m_2...m\_L$ di lunghezza L disposto in una matrice rettangolare di r righe × k colonne, dove:
- k = caratteri per avvolgimento (determinato dalla circonferenza del cilindro e dalla dimensione del carattere)
- r = ⌈L/k⌉ (la funzione ceiling, arrotonda per eccesso)

La matrice M_plain viene costruita scrivendo il messaggio per righe:

M_plain[i,j] = m_{i×k+j} per i ∈ {0,...,r-1}, j ∈ {0,...,k-1}

La cifratura consiste nel leggere per colonne invece che per righe. Formalmente, definiamo la permutazione π:

π: {0,1,...,L-1} → {0,1,...,L-1}

π(i) = (i mod r) × k + ⌊i/r⌋

dove i è la posizione nel testo chiaro e π(i) la posizione nel testo cifrato.

Questa permutazione è biettiva (quindi invertibile), con la funzione inversa:

$\pi^{-1}(j)$ = (j mod k) × r + ⌊j/k⌋

La correttezza è verificabile direttamente: $\pi^{-1}(\pi(i)) = i$ per ogni i.

3.4 Un Esempio possibile: L'Attacco all'Alba

Consideriamo il messaggio greco "ΕΠΙΤΙΘΕΣΘΑΙΤΗΝΑΥΓΗΝ" (Epitithethai tēn augēn - "Attaccare all'alba"), con L = 20 caratteri e un cilindro con k = 8 caratteri per giro.
La matrice del testo chiaro (r = ⌈20/8⌉ = 3 righe):

```
      j=0  j=1  j=2  j=3  j=4  j=5  j=6  j=7
i=0   Ε    Π    Ι    Τ    Ι    Θ    Ε    Σ
i=1   Θ    Α    Ι    Τ    Η    Ν    Α    Υ
i=2   Γ    Η    Ν
```

(indica il padding per completare la matrice)

La lettura per colonne (la cifratura):

- Colonna 0: Ε, Θ, Γ
- Colonna 1: Π, Α, Η
- Colonna 2: Ι, Ι, Ν
- Colonna 3: Τ, Τ,
- Colonna 4: Ι, Η,
- Colonna 5: Θ, Ν,
- Colonna 6: Ε, Α,
- Colonna 7: Σ, Υ,

Il testo cifrato: ΕΘΓΠΑΗΙΙΝΤΙΗΘΝΕΑΣΥ
Rimuovendo il padding: ΕΘΓΠΑΗΙΙΝΤΙΗΘΝΕΑΣΥ

3.5 L'Analisi di Sicurezza: La Forza e la Debolezza

Lo spazio delle chiavi: In teoria, il diametro d del cilindro è un parametro continuo ($d \in \mathbb{R}_+$). In pratica, le limitazioni manifatturiere dell'antichità – gli utensili in bronzo, la lavorazione manuale del legno – imponevano una tolleranza di circa ±0,5 mm. Assumendo un range realistico $d \in [2{,}0; 5{,}0]$ cm con gli incrementi di 0,5 mm, lo spazio delle chiavi effettivo è circa $K \approx 60$.
La vulnerabilità a forza bruta: Con solo 60 possibilità, un attaccante potrebbe testare tutti i diametri in circa 30 minuti (assumendo 30 secondi per un test manuale). Il sistema è dunque vulnerabile all'attacco esaustivo anche con la tecnologia antica.
La resistenza all'analisi di frequenza: Qui lo scitale mostra la sua forza. Essendo una trasposizione pura, preserva le frequenze dei caratteri:

$$f_{cifrato}(\alpha) = f_{chiaro}(\alpha) \quad \forall \alpha \in A$$

L'analisi di frequenza monocarattere – devastante contro i cifrari di sostituzione come vedremo – è completamente inefficace contro le trasposizioni pure.
La vulnerabilità ai pattern: Tuttavia, se l'attaccante conosce che il sistema è una trasposizione colonnare e può stimare k, emerge un attacco basato sui pattern di digrammi. I digrammi comuni nella lingua originale (nel greco: ΤΟ, ΤΗ, ΚΑ) ora appaiono separati da una distanza k nel testo cifrato. Identificando sistematicamente i digrammi frequenti a una distanza fissa, si può dedurre k e ricostruire la matrice.
La complessità di questo attacco è $O(L^2)$ per l'identificazione dei digrammi più $O(k_{max})$ per il test dei valori k – fattibile anche manualmente per i testi di lunghezza moderata.

3.6 L'Utilizzo Storico: Lisandro e Farnabazo

Plutarco, nella sua Vita Lysandri (19.3-5), riporta un episodio emblematico. Nel 395 a.C., Lisandro comandava le forze spartane in Asia Minore durante le operazioni

contro il satrapo persiano Farnabazo. Gli efori a Sparta, avendo deciso di richiamare Lisandro per le ragioni politiche, inviarono l'ordine cifrato via scitale.
Il messaggero fu intercettato dagli agenti persiani. Trovando solo una striscia di cuoio con le lettere apparentemente casuali, gli agenti non poterono estrarre l'intelligence utile. Il messaggio raggiunse Lisandro, che lo decifrò con il proprio scitale, permettendo l'esecuzione ordinata del richiamo senza che i Persiani anticipassero i movimenti spartani.
Il successo fu dovuto a:
- Il messaggero ignorava il contenuto (security by ignorance)
- Gli intercettatori erano privi dello scitale corretto
- Il tempo limitato per la crittanalisi (la situazione tattica dinamica)
- La mancanza di cribs (il contenuto sconosciuto agli intercettatori)
Lo scitale spartano rappresenta dunque il primo sistema documentato di trasposizione algoritmica, segnando l'emergere del concetto di una chiave fisica separata dal metodo – un passo cruciale verso la crittografia matura.

CAPITOLO 4

Il Cifrario di Cesare: La Sostituzione Sistematica

Con Gaio Giulio Cesare entriamo nel territorio della sostituzione monoalfabetica sistematica – un salto concettuale che dominerà la crittografia per il successivo millennio. Mentre lo scitale riarrangiava le posizioni, il cifrario di Cesare trasforma le identità stesse dei caratteri secondo una regola matematica semplice ma potente.

4.1 La Testimonianza Storica e il Contesto

La nostra fonte primaria è Gaio Svetonio Tranquillo, nella sua De Vita Caesarum (Libro I, Divus Julius, capitolo 56), scritta circa nel 121 d.C., quasi due secoli dopo la morte di Cesare. Svetonio scrive:
"Extant et ad Ciceronem, item ad familiares domesticis de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum verbum effici posset: quae si quis investigare et persequi velit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet."
"Esistono anche le lettere a Cicerone e agli amici su le questioni personali, nelle quali, se c'era qualcosa di particolare riservatezza da trasmettere, scriveva in cifra, cioè disponendo le lettere in modo tale che non si potesse formare alcuna parola: per scoprirne il significato bisogna sostituire la quarta lettera dell'alfabeto, cioè D al posto di A, e così via per le altre."
Cesare operava in un contesto militare e politico straordinariamente complesso: le campagne galliche (58-50 a.C.), la guerra civile contro Pompeo (49-45 a.C.), le comunicazioni con i generali lontani (Labieno in Gallia, Marco Antonio in Italia,

Cicerone per le questioni politiche). I dispacci dovevano attraversare centinaia di chilometri di territorio ostile, portati dai messaggeri vulnerabili all'intercettazione.

4.2 La Formalizzazione Matematica: Il Gruppo delle Traslazioni

Il cifrario di Cesare rappresenta il caso più semplice del cifrario affine, definibile formalmente come segue. Sia A = {A, B, C, ..., Z} l'alfabeto latino classico con A = 23 lettere (senza J, U, W che non esistevano nel latino arcaico). Definiamo la funzione di codifica numerica:

$\phi: A \rightarrow \mathbb{Z}_{23}$

dove $\phi(A) = 0$, $\phi(B) = 1$, ..., $\phi(Z) = 22$, e la funzione inversa $\phi^{-1}: \mathbb{Z}_{23} \rightarrow A$.

Il cifrario di Cesare con il parametro k (lo shift) è definito da:

La funzione di cifratura: $E_k: \mathbb{Z}_{23} \rightarrow \mathbb{Z}_{23}$

$E_k(x) \equiv x + k \pmod{23}$

La funzione di decifratura: $D_k: \mathbb{Z}_{23} \rightarrow \mathbb{Z}_{23}$

$D_k(y) \equiv y - k \pmod{23}$

La correttezza è immediata:

$D_k(E_k(x)) = D_k(x + k) = (x + k) - k \equiv x \pmod{23}$ ✓

La struttura algebrica: L'insieme $\{E_k \quad k \in \mathbb{Z}_{23}\}$ forma un gruppo abeliano sotto la composizione delle funzioni:

- La chiusura: $E_{k_1} \circ E_{k_2} = E_{\{(k_1+k_2) \bmod 23\}}$
- L'associatività: La composizione è associativa
- L'elemento identità: E_0 (nessuna trasformazione)
- L'elemento inverso: $E_k^{-1} = E_{\{(-k) \bmod 23\}} = D_k$
- La commutatività: $E_{k_1} \circ E_{k_2} = E_{k_2} \circ E_{k_1}$

L'ordine del gruppo è G = 23 (un numero primo), quindi il gruppo è ciclico e ogni elemento non-identità è un generatore.

4.3 Il Sistema di Cesare Storico (k=3)

Svetonio specifica che Cesare utilizzava uno shift di 3 posizioni. La tabella di sostituzione risultante:

```
Chiaro:  A B C D E F G H I K L M N O P Q R S T V X Y Z
Cifrato: D E F G H I J K L N O P Q R S T U V W Y Z A B
```

(Nota: J rappresenta qui lo shift da I, mantenendo la consistenza con l'alfabeto a 23 lettere)

Un esempio storico ricostruito: Il messaggio "CAESAR VENIT" (Cesare arriva) – un possibile dispaccio urgente dal campo di battaglia.

L'applicazione carattere per carattere:

- C: $\phi(C) = 2 \rightarrow E_3(2) = 5 \rightarrow \phi^{-1}(5) = F$
- A: $\phi(A) = 0 \rightarrow E_3(0) = 3 \rightarrow \phi^{-1}(3) = D$
- E: $\phi(E) = 4 \rightarrow E_3(4) = 7 \rightarrow \phi^{-1}(7) = H$
- S: $\phi(S) = 17 \rightarrow E_3(17) = 20 \rightarrow \phi^{-1}(20) = V$
- A: → D
- R: $\phi(R) = 16 \rightarrow E_3(16) = 19 \rightarrow \phi^{-1}(19) = T$
- Spazio:
- V: $\phi(V) = 19 \rightarrow E_3(19) = 22 \rightarrow \phi^{-1}(22) = Z$
- E: → H
- N: $\phi(N) = 12 \rightarrow E_3(12) = 15 \rightarrow \phi^{-1}(15) = P$
- I: $\phi(I) = 8 \rightarrow E_3(8) = 11 \rightarrow \phi^{-1}(11) = L$
- T: $\phi(T) = 18 \rightarrow E_3(18) = 21 \rightarrow \phi^{-1}(21) = X$

Il testo cifrato: FDHVDT ZHPLX

L'esempio con il wrap-around (il comportamento modulare) è particolarmente istruttivo:

- X: $\phi(X) = 20 \rightarrow E_3(20) = 23 \bmod 23 = 0 \rightarrow \phi^{-1}(0) = A$
- Y: $\phi(Y) = 21 \rightarrow E_3(21) = 24 \bmod 23 = 1 \rightarrow \phi^{-1}(1) = B$
- Z: $\phi(Z) = 22 \rightarrow E_3(22) = 25 \bmod 23 = 2 \rightarrow \phi^{-1}(2) = C$

Quindi "XYZ" → "ABC", dimostrando perfettamente la natura ciclica della trasformazione – un concetto che diverrà centrale nei cifrari polialfabetici successivi.

4.4 La Fragilità Intrinseca: Lo Spazio delle Chiavi Minuscolo

Il tallone d'Achille del cifrario di Cesare è immediatamente evidente: lo spazio delle chiavi K = {0, 1, 2, ..., 22} ha la dimensione K = 23. Escludendo k=0 (l'identità, nessuna cifratura), abbiamo solo K_effettivo = 22 chiavi significative.
L'attacco a forza bruta: La procedura è banale:

```
Per ogni k ∈ {1, 2, ..., 22}:
   Applicare D_k al testo cifrato
   Verificare se il risultato è latino leggibile
   Se sì, terminare (la chiave trovata)
```

Con un test manuale di circa 1 minuto per tentativo, il tempo medio è:

$T_{medio} = K/2 \approx 11$ minuti

$T_{max} = 22$ minuti

4.5 L'Analisi di Frequenza: La Spada di Damocle

La debolezza più profonda del cifrario di Cesare non è lo spazio delle chiavi ristretto, ma la sua natura di sostituzione monoalfabetica. Ogni lettera del testo chiaro è sempre mappata nella stessa lettera del testo cifrato, preservando intatte tutte le proprietà statistiche della lingua:

- Le frequenze dei caratteri: Se E appare con la frequenza del 12% nel latino, la sua cifratura (H con k=3) apparirà con la frequenza del 12% nel testo cifrato
- I pattern di ripetizione: Le parole ripetute nel chiaro producono i pattern ripetuti nel cifrato
- La lunghezza del testo: Completamente preservata (la funzione biunivoca)

Le frequenze nel latino classico, estratte da un corpus estensivo di Cicerone, Cesare, e Virgilio (circa 500.000 caratteri), mostrano una distribuzione marcatamente non uniforme:

Lettera	Frequenza (%)	Lettera	Frequenza (%)
E	12,02	P	2,51
I	11,03	B	2,15
A	9,68	G	1,98
T	8,77	F	1,92
S	7,93	V	1,89

N	7,12	Q	1,78
R	6,89	H	1,23
V	6,43	Y	0,67
M	5,11	K	0,52
C	4,35	X	0,48
O	3,77	Z	0,11
L	3,21		

La procedura di crittanalisi per frequenza:

1. Calcolare le frequenze nel testo cifrato: f_C(c) per ogni c ∈ A
2. Identificare il carattere più frequente: c_max = argmax{f_C(c)}
3. Ipotizzare c_max ← E (la lettera più frequente nel latino)
4. Calcolare lo shift ipotizzato: k_ipotizzato = ϕ(c_max) - ϕ(E) mod 23
5. Applicare D_k_ipotizzato e verificare la leggibilità

Un esempio quantitativo: Dato un testo cifrato di 1000 caratteri con le frequenze osservate:

- H: 121 occorrenze (12,1%)
- L: 110 occorrenze (11,0%)
- D: 97 occorrenze (9,7%)

L'ipotesi: H ← E (vicino alla frequenza attesa 12,02%)

k = ϕ(H) - ϕ(E) = 7 - 4 = 3

Questo è esattamente lo shift storico di Cesare! Un singolo calcolo, senza il bisogno di testare le 22 possibilità.

La probabilità di successo per un testo di lunghezza L > 100 con la distribuzione di frequenze standard latina è P ≈ 0,92 – il metodo è estremamente affidabile.

4.6 L'Indice di Coincidenza: Una Firma Statistica

Un concetto che diverrà cruciale nei capitoli successivi merita un'introduzione qui: l'Indice di Coincidenza (IC), sviluppato formalmente da William Friedman nel 1920 ma osservabile implicitamente già nell'analisi classica.

Per un testo T di lunghezza N, con n_i occorrenze della lettera i:

$IC(T) = \Sigma_{i=o}^{22} [n_i(n_i-1)] / [N(N-1)]$

L'IC rappresenta la probabilità che due lettere scelte casualmente dal testo siano identiche. Per il latino con la distribuzione naturale:

$IC\_latino \approx \Sigma_i f_i^2 \approx 0,0667$

dove f_i sono le frequenze relative. Per un testo casuale:
$IC\_casuale = 1/23 \approx 0,0435$

La proprietà critica: Il cifrario di Cesare preserva l'IC:

$IC\_cifrato = IC\_latino \approx 0,0667$

Questo permette l'identificazione immediata come una sostituzione monoalfabetica. Un testo cifrato con IC ≈ 0,0667 non può essere prodotto da un cifrario polialfabetico (che abbassa l'IC verso la distribuzione uniforme), ma deve essere una sostituzione monoalfabetica – un'informazione che restringe drasticamente lo spazio di ricerca per il crittoanalista.

4.7 Le Varianti e le Generalizzazioni

Il Cifrario di Augusto: Svetonio (Divus Augustus, 88) riporta che Augusto utilizzava lo shift k=1 con un'eccezione peculiare: la lettera Z non veniva mappata ad A (come richiesto dalla coerenza modulare) ma a "ZZ" (il raddoppio). Questa irregolarità:

- Introduce la non-uniformità nella mappatura
- Causa l'espansione del testo (Z → ZZ aggiunge i caratteri)
- Crea un marker riconoscibile (la presenza di ZZ nel cifrato)

Paradossalmente, la "complessità aggiunta" riduce la sicurezza: ZZ diventa un segnale distintivo del sistema di Augusto, permettendo l'identificazione immediata e facilitando l'attacco.

La generalizzazione. Il Cifrario Affine: il Cesare è un caso speciale (a=1) del cifrario affine generale:

$E\_(a,b)(x) \equiv ax + b \pmod n$

dove:

- $a \in \mathbb{Z}\_n$ (le unità di $\mathbb{Z}\_n$, cioè $\gcd(a,n) = 1$)
- $b \in \mathbb{Z}\_n$
- $n = A$

La condizione di invertibilità: gcd(a,n) = 1 è necessaria e sufficiente affinché E_(a,b) sia una biezione.

La funzione di decifratura:
$D_{(a,b)}(y) \equiv a^{-1}(y - b) \pmod{n}$

dove a^{-1} è l'inverso moltiplicativo di a modulo n.

Lo spazio delle chiavi: $K = \phi(n) \times n$, dove ϕ è la funzione totiente di Eulero. Per n=23 (primo):

$\phi(23) = 22$

$K = 22 \times 23 = 506$

Un sostanziale miglioramento rispetto al Cesare (23 chiavi), ma ancora vulnerabile all'analisi di frequenza dato che rimane una sostituzione monoalfabetica.
Il cifrario di Cesare, per quanto crittograficamente debole, rappresenta una pietra miliare concettuale: la prima implementazione documentata di una trasformazione algoritmica con un parametro separabile (la chiave k). Questo concetto della separazione tra l'algoritmo e la chiave diverrà fondamentale per tutta la crittografia successiva, culminando nel principio di Kerckhoffs (XIX secolo): "La sicurezza deve risiedere nella chiave, non nel metodo."

CAPITOLO 5

Al-Kindī e la Nascita della Crittanalisi Scientifica

Il millennio che separa la caduta di Roma (476 d.C.) dall'inizio del Rinascimento europeo (XIV secolo) è spesso erroneamente chiamato "età oscura" – un'etichetta che ignora completamente il fiorire straordinario della scienza e della matematica nel mondo islamico. Mentre l'Europa occidentale frammentata faticava a preservare anche i rudimenti del sapere classico, il califfato abbaside a Baghdad diventava il faro di conoscenza, ospitando la più grande concentrazione di studiosi, matematici e filosofi dal tempo di Alessandria ellenistica.
È in questo contesto che nasce la crittanalisi come una disciplina scientifica rigorosa, attraverso il lavoro di un poliedrico genio del IX secolo.

5.1 Baghdad: La Casa della Sapienza

Il califfato abbaside (750-1258 d.C.), con la capitale a Baghdad, rappresentò l'apogeo della civiltà islamica classica. Il califfo Hārūn al-Rashīd (786-809), immortalato nelle Mille e Una Notte, fondò la Bayt al-Ḥikma (la Casa della Sapienza), inizialmente come una biblioteca e un centro di traduzione, espansa massicciamente dal suo successore al-Ma'mūn (813-833) in una vera e propria accademia multidisciplinare.
Qui, i testi greci, persiani, sanscriti venivano tradotti in arabo; i matematici sviluppavano l'algebra (dall'arabo al-jabr); gli astronomi raffinavano i modelli planetari; i medici componevano i trattati che avrebbero dominato la medicina europea per i secoli. L'atmosfera intellettuale era di una sintesi creativa – combinare l'eredità classica con le innovazioni originali, guidati dal metodo scientifico empirico piuttosto che dall'autorità dottrinale.

5.2 Abū Yūsuf Ya'qūb ibn Isḥāq al-Kindī (801-873 d.C.)

Al-Kindī, nato a Kufa da una famiglia aristocratica araba, incarnò l'ideale del ḥakīm – il saggio universale. Soprannominato "Philosophus Arabum" (Il Filosofo degli Arabi) dai traduttori latini medievali, produsse oltre 260 opere coprendo:

- La filosofia: i commentari su Aristotele e Platone; la metafisica; l'epistemologia
- La matematica: la geometria, la teoria dei numeri, il calcolo indiano
- L'ottica: un trattato sulla prospettiva che influenzerà Roger Bacon
- La medicina: la farmacologia, il dosaggio matematico dei farmaci
- La musica: la teoria musicale, l'acustica
- La crittografia: la Risālah fī Istikhrāj al-Mu'ammā (Il Manoscritto sulla Decifratura dei Messaggi Criptografici)

Quest'ultima opera, composta circa nell'850 d.C., è straordinaria non solo per il contenuto tecnico ma per l'approccio metodologico: al-Kindī non propone semplicemente dei trucchi pratici, ma sviluppa una teoria generale basata sull'osservazione empirica e sul ragionamento matematico.
Il manoscritto, preservato nella Süleymaniye Library di Istanbul (MS Aya Sofya 4830), rimase sconosciuto al mondo occidentale fino alla pubblicazione moderna nel 1987. Eppure, il suo contenuto anticipa di quasi mille anni i metodi che l'Europa avrebbe "scoperto" solo nel XIX secolo.

5.3 Il Principio Fondamentale: La Regolarità Statistica del Linguaggio

L'intuizione centrale di al-Kindī è profonda nella sua semplicità: ogni lingua naturale possiede un'impronta statistica caratteristica – le frequenze relative delle lettere e le loro combinazioni che rimangono notevolmente stabili attraverso i testi diversi. Questa regolarità è una proprietà intrinseca della struttura linguistica, riflettendo la fonologia, la morfologia, e la sintassi della lingua.

Al-Kindī scrive (traduzione dal manoscritto):
"Un modo per risolvere un messaggio cifrato, se sappiamo in quale lingua è stato scritto, è procurarci un altro testo in chiaro nella stessa lingua, lungo abbastanza da riempire circa una pagina, e quindi contare le occorrenze di ogni lettera. Chiamiamo la lettera più frequente la 'prima', la seconda più frequente la 'seconda', e così via, finché consideriamo tutte le diverse lettere nel testo in chiaro. Poi guardiamo al testo cifrato che vogliamo decifrare e classifichiamo similmente i suoi simboli. Troviamo il simbolo più comune e lo sostituiamo con la 'prima' lettera del testo in chiaro, il secondo simbolo più comune con la 'seconda' lettera, e così via, finché rappresentiamo tutti i simboli del testo cifrato con le lettere del testo in chiaro."

5.4 La Formalizzazione Matematica della Stabilità Statistica

Possiamo rendere l'intuizione di al-Kindī matematicamente precisa. Sia T un testo nella lingua L, di lunghezza N caratteri, su un alfabeto A. Per ogni lettera $\alpha \in A$:

La frequenza assoluta: $n(\alpha)$ = il numero delle occorrenze di α in T

La frequenza relativa: $f(\alpha) = n(\alpha) / N$

La distribuzione di frequenza della lingua è:

$F\_L = \{(\alpha, f(\alpha)) \ \alpha \in A\}$

Il claim empirico di al-Kindī è che F_L è invariante (stazionaria) rispetto a T: per i testi sufficientemente lunghi nella stessa lingua, le frequenze relative convergono ai valori caratteristici.

Formalmente, per una sequenza di testi $T_1, T_2, ..., T\_k$ ciascuno di lunghezza $N \to \infty$:

$\lim\_(N\to\infty) f\_T(\alpha) \to f\_L(\alpha)$

con la varianza decrescente come $O(1/\sqrt{N})$ (il teorema del limite centrale).

Per il latino classico (un corpus di ~500.000 caratteri), le frequenze convergono a:

$f\_L(E) \approx 0{,}1202$, $f\_L(I) \approx 0{,}1103$, $f\_L(A) \approx 0{,}0968$, ...

con la deviazione standard $\sigma < 0{,}005$ per i testi di lunghezza > 1000 caratteri.

5.5 L'Invarianza per la Sostituzione Monoalfabetica

La proprietà cruciale sfruttata dall'analisi di frequenza è:

Teorema (La Preservazione Statistica): Sia E: A → A una sostituzione monoalfabetica (una biezione). Sia P un testo chiaro con la distribuzione F_P, C = E(P) il testo cifrato con la distribuzione F_C. Allora:

$f_C(E(\alpha)) = f_P(\alpha)\ \forall \alpha \in A$

Dimostrazione: E è una biezione, quindi ogni occorrenza di α in P viene mappata in E(α) in C esattamente una volta:

$n_C(E(\alpha)) = \{i\ \ C[i] = E(\alpha)\} = \{i\ \ E(P[i]) = E(\alpha)\} = \{i\ \ P[i] = \alpha\} = n_P(\alpha)$

(l'uguaglianza cruciale usa l'iniettività di E)

Quindi: $f_C(E(\alpha)) = n_C(E(\alpha))/N = n_P(\alpha)/N = f_P(\alpha)$ ✓
La conseguenza crittanalitica: Le frequenze dei caratteri nel testo cifrato rivelano direttamente le frequenze del testo chiaro, semplicemente "rimappate" attraverso E. Se identifichiamo quale carattere cifrato corrisponde a quale carattere chiaro (attraverso il matching delle frequenze), ricostruiamo E^{-1} e quindi il messaggio originale.

5.6 L'Analisi di Frequenza dell'Arabo Classico

Al-Kindī, lavorando con i testi arabi, necessitava della distribuzione di frequenza dell'arabo classico del IX secolo. Il Corano, il testo sacro immutabile e universalmente disponibile, forniva il corpus ideale: 77.797 parole, circa 330.000 caratteri, lo stile stilizzato ma rappresentativo.

Le frequenze dell'arabo coranico (la traslitterazione semplificata):

Lett.	Traslitt.	Freq. (%)	Lett.	Traslitt.	Freq. (%)
ا	ā (alif)	12,81	ع	(ayn.)	3,12
ل	l (lām)	9,98	ف	f (fā)	2,98
ن	n (nūn)	7,12	ق	q (qāf)	2,76
م	m (mīm)	6,23	ك	k (kāf)	2,54
و	w/ū (wāw)	5,86	د	d (dāl)	2,31
ي	y/ī (yā)	5,67	ه	h (hā)	2,15
ر	r (rā)	4,89	س	s (sīn)	1,98
ت	t (tā)	4,11	حṭ	(ḥā)	1,87
ب	b (bā)	3,44	...	...	

Notare la distribuzione simile al latino/greco: poche lettere dominano (alif, lām, nūn rappresentano ~30%), molte lettere rare. Questa ineguaglianza è sfruttabile.

5.7 La Procedura Completa di Crittanalisi

Al-Kindī non si limita all'analisi monocarattere, ma descrive un approccio multilivello progressivamente più sofisticato:

Livello 1: L'Analisi Monocarattere

1. Calcolare le frequenze f_C(c) per ogni c nel testo cifrato
2. Ordinare per la frequenza decrescente: c_1, c_2, ..., c_n
3. Confrontare con le frequenze note della lingua: l_1, l_2, ..., l_n
4. Ipotizzare $c_1 \leftarrow l_1$, $c_2 \leftarrow l_2$, ...
5. Costruire la mappatura D_ipotizzata: A_cifrato → A_chiaro
6. Applicare e verificare la leggibilità

Livello 2: L'Analisi dei Bigrammi (i Digrafi)

Se l'analisi monocarattere produce i risultati ambigui, al-Kindī suggerisce di esaminare le coppie di lettere consecutive. Ogni lingua ha i bigrammi caratteristici:
- Arabo classico: AL (ال, l'articolo definito), AN (ان, "che"), MA (ما, "cosa"), LA (لا, "no")
- Latino: QU, IN, NT, ER, ES
- Greco: TH, HE, ON, TO

La procedura:

1. Estrarre tutti i bigrammi dal testo cifrato: B_C = {C[i]C[i+1]}
2. Calcolare le frequenze: f_B(b) per ogni b
3. Identificare i bigrammi cifrati più frequenti
4. Confrontare con i bigrammi noti della lingua
5. Risolvere il sistema di equazioni per la mappatura dei caratteri

Se il bigramma cifrato "TV" appare molto frequentemente e ipotizziamo TV ← QU (latino), allora T ← Q e V ← U. Verificare la consistenza con gli altri bigrammi contenenti T o V rafforza o smentisce l'ipotesi.

Livello 3: I Pattern Strutturali

- Le parole corte: La lingua ha le parole di 1-3 lettere con la frequenza alta (gli articoli, le preposizioni, le congiunzioni)
- I pattern ripetuti: Le parole comuni ripetute producono i pattern riconoscibili

- I vincoli grammaticali: Certe sequenze impossibili nella lingua (es. tre consonanti consecutive nel latino classico sono rare)

L'Algoritmo Iterativo Completo:

Input: il testo_cifrato C, la distribuzione_lingua F_L
Output: il testo_chiaro P

1. mappatura ← {} // il dizionario vuoto
2. fiducia ← array[A] inizializzato a 0
3. // La Fase monocarattere
4. Per ogni c in A_cifrato:
5. f_c ← frequenza(c, C)
6. l ← lettera_più_simile(f_c, F_L)
7. mappatura[c] ← l
8. fiducia[c] ← similarità(f_c, F_L[l])

9. // Il Raffinamento con i bigrammi
10. Per ogni bigramma b = c_1c_2 in C con la freq. alta:
11. b_candidati ← bigrammi_simili(frequenza(b, C), F_L)
12. Per ogni (l_1, l_2) in b_candidati:
13. Se consistente con mappatura[c_1], mappatura[c_2]:
14. aumenta fiducia[c_1], fiducia[c_2]
15. Altrimenti se fiducia[c_1] < soglia:
16. considera la revisione mappatura[c_1] ← l_1

17. // La Verifica e la correzione
18. P ← applica_mappatura(C, mappatura)
19. score ← valuta_leggibilità(P) // basato sul dizionario, la grammatica
20. Se score < soglia_accettabile:
21. identifica_errori(P) // le parole quasi-corrette
22. correggi_mappatura_locale()
23. goto 18

24. Return P

5.8 La Validazione Statistica: Il Test Chi-Quadrato

Al-Kindī non possedeva la teoria statistica formale (che emergerà solo nel XIX-XX secolo), ma il suo metodo implica un'intuizione profonda: una mappatura corretta dovrebbe produrre un testo con la distribuzione vicina alla lingua naturale.
Formalizziamo questo con il test χ^2 (chi-quadrato), sviluppato da Karl Pearson (1900) ma applicabile retroattivamente:

Per testare se la distribuzione osservata f_oss coincide con la distribuzione attesa f_att:
$\chi^2 = \Sigma_{i=1}^{n} [(O_i - E_i)^2 / E_i]$

dove:
- O_i = la frequenza osservata della lettera i (il conteggio effettivo)
- E_i = la frequenza attesa della lettera i (N × f_att(i))
- n = A (la dimensione dell'alfabeto)

I gradi di libertà: $\nu = n - 1$

L'ipotesi nulla: H_o: il testo osservato proviene dalla lingua ipotizzata

Per $\alpha = 0,05$ (il livello di significatività 95%), n = 23 (latino):
χ^2_critico(22, 0,05) = 33,92

La decisione:

- Se χ^2_calcolato < 33,92: non rifiutare H_o (la distribuzione compatibile con il latino)
- Se χ^2_calcolato > 33,92: rifiutare H_o (la distribuzione anomala, la mappatura probabilmente errata)

Un esempio numerico: Un testo decifrato candidato, N = 1000 caratteri

Lettera	Oss. (O)	Atteso (E)	$(O-E)^2/E$
E	118	120,2	0,040
I	107	110,3	0,099
A	99	96,8	0,050
T	85	87,7	0,083
S	81	79,3	0,036
N	73	71,2	0,045
R	68	68,9	0,012
...	...	...	...

Sommando tutti i contributi: $\chi^2 \approx 18,47$

Poiché 18,47 < 33,92, la distribuzione è compatibile con il latino. La mappatura è probabilmente corretta.

5.9 L'Indice di Coincidenza: La Misura della "Naturalità"
Un altro strumento statistico potente (formalizzato da William Friedman nel 1920 ma osservabile empiricamente già in al-Kindī) è l'Indice di Coincidenza (IC), introdotto nel capitolo precedente.

Ricordiamo la definizione:

$IC(T) = \Sigma_{i=o}^{n-1} [n_i(n_i-1)] / [N(N-1)]$

L'interpretazione probabilistica: la probabilità che due lettere scelte casualmente dal testo siano identiche.

I valori caratteristici:

- La lingua naturale (latino, arabo, greco): IC ≈ 0,065-0,075
- Il testo casuale (la distribuzione uniforme): IC = 1/n ≈ 0,038-0,043

La proprietà cruciale per la crittanalisi:

Teorema (L'Invarianza dell'IC per la Sostituzione Monoalfabetica): Per qualsiasi sostituzione monoalfabetica E:

IC(E(T)) = IC(T)

Dimostrazione: Già fornita nel Capitolo 4. La chiave è che E, essendo una biezione, preserva le molteplicità: se la lettera α appare n volte in T, allora E(α) appare esattamente n volte in E(T).

L'applicazione diagnostica:

- Calcolare l'IC del testo cifrato
- Se IC ≈ IC_lingua: probabile sostituzione monoalfabetica (come il Cesare)
- Se IC ≈ IC_casuale: probabile sostituzione polialfabetica o un altro metodo più complesso

5.10 I Raffinamenti: L'Analisi Contestuale e i Pattern Linguistici

Al-Kindī dimostra una sofisticazione ulteriore considerando i vincoli linguistici oltre alle pure statistiche:

I vincoli fonotattici: Certe sequenze di consonanti/vocali sono impossibili o rare. Nell'arabo, tre consonanti consecutive senza una vocale intermedia sono estremamente rare. Se la decifrazione produce tali sequenze, è probabilmente errata.
La morfologia: Le radici arabe seguono i pattern (CvCvC dove C=consonante, v=vocale). Riconoscere i pattern radicali può aiutare a identificare le lettere ambigue.

La sintassi: L'ordine delle parole (VSO nell'arabo classico: Verbo-Soggetto-Oggetto) e le particelle grammaticali obbligatorie forniscono i vincoli posizionali.

Un esempio pratico: Se la decifrazione produce la sequenza "...AL...AL...AL..." ripetuta frequentemente, e AL (ال) è l'articolo definito arabo, possiamo:

- Confermare che A ← ا e L ← ل nella mappatura
- Usare questa certezza per raffinare le ipotesi su altre lettere
- Cercare il pattern "AL + sostantivo" per identificare ulteriori sostantivi comuni

Questo approccio multilivello – statistico, linguistico, contestuale – rappresenta l'essenza della crittanalisi moderna. Al-Kindī, nel IX secolo, aveva già compreso che violare i cifrari richiede la sintesi di matematica, linguistica, e ragionamento logico sistematico.

5.12 Le Lezioni Epistemologiche: Il Metodo Scientifico nella Crittanalisi

Oltre al contenuto tecnico, l'approccio di al-Kindī offre delle lezioni metodologiche profonde:

L'empirismo: Basa la teoria sull'osservazione sistematica (il conteggio delle frequenze nei corpus estesi), non sulla speculazione astratta

La generalizzazione: Cerca i principi universali (la regolarità statistica del linguaggio) piuttosto che i trucchi ad hoc

La testabilità: Il metodo produce le predizioni verificabili (se la mappatura è corretta, il testo risultante dovrebbe essere leggibile)

L'iterazione: Riconosce che il processo è iterativo – le ipotesi iniziali vengono raffinate attraverso i test successivi

Questo è il metodo scientifico maturo, applicato a un dominio non tradizionalmente considerato "scientifico". Al-Kindī dimostra che la crittanalisi non è un'arte oscura o la magia, ma una disciplina razionale governata dai principi matematici e linguistici comprensibili e sistematizzabili.
La sua eredità non è solo tecnica ma epistemologica: stabilisce la crittanalisi come una scienza, aprendo la strada agli sviluppi che culmineranno a Bletchley Park un millennio dopo.

CAPITOLO 6

Il Rinascimento della Crittografia: Leon Battista Alberti

L'Italia del Quattrocento fu il teatro di una straordinaria fioritura culturale e intellettuale. Mentre le signorie e le repubbliche competevano ferocemente per il potere e l'influenza – con le alleanze mutevoli, i complotti incessanti, e le guerre endemiche – la diplomazia divenne un'arte raffinatissima. E con essa, la necessità delle comunicazioni sicure raggiunse un'urgenza senza precedenti.

6.1 Il Contesto Rinascimentale: La Diplomazia e il Segreto

La frammentazione politica dell'Italia – il Ducato di Milano, la Repubblica di Venezia, la Repubblica di Firenze, lo Stato Pontificio, il Regno di Napoli, più le dozzine di entità minori – creò un sistema di equilibrio precario. Ogni corte manteneva una rete estensiva di ambasciatori, informatori, spie. Le comunicazioni diplomatiche attraversavano i territori ostili, soggette all'intercettazione sistematica.

Le cifre diplomatiche divennero uno strumento essenziale. Ogni cancelleria sviluppava i propri sistemi, custoditi gelosamente. La figura del segretario cifrato – il funzionario responsabile per la cifratura e la decifratura della corrispondenza sensibile – acquisì un'importanza cruciale. Alcuni, come Giovanni Soro a Venezia (fine del XV secolo), raggiunsero la fama (e il timore) come i crittoanalisti capaci di violare le cifre straniere.

In questo ambiente competitivo e intellettualmente fertile emerge Leon Battista Alberti.

6.2 Leon Battista Alberti (1404-1472): L'Uomo Universale

Se al-Kindī incarnò l'ideale del ḥakīm islamico, Alberti incarnò perfettamente l'ideale rinascimentale dell'uomo universale: architetto, teorico d'arte, matematico, filosofo, crittografo, poeta, drammaturgo, atleta (!) – eccellendo in ogni campo intrapreso.

L'opera crittografica:

"De componendis cifris et litterarum notis ad omnem linguam opportunis" (Sulla composizione delle cifre e dei segni delle lettere adatti a ogni lingua), dedicato a Leonardo Dato, il segretario pontificio, scritto nel 1467.

Questo trattato di circa 25 pagine (nell'edizione moderna) è rivoluzionario: il primo testo occidentale a discutere la crittografia sistematicamente, anticipando di decenni gli sviluppi che domineranno il XVI secolo.

6.3 Il Disco Cifrante: L'Innovazione Meccanica

L'invenzione più celebre di Alberti è il disco cifrante (formula): un dispositivo fisico che meccanizza il processo di cifratura polialfabetica. Concettualmente semplice, le sue implicazioni sono profonde.
La descrizione fisica (le ricostruzioni moderne dalle descrizioni nel trattato):

Il disco esterno (fisso): Diametro ~10-15 cm, diviso in 24 settori uguali

- Contenuto: 20 lettere maiuscole latine (senza H, J, K, Y) + 4 cifre (1, 2, 3, 4)
- Sequenza: A B C D E F G I L M N O P Q R S T V X Z 1 2 3 4

Il disco interno (mobile): Diametro leggermente minore, concentrico al primo

- Contenuto: l'alfabeto cifrato in lettere minuscole
- Sequenza: a b c d e f g h i k l m n o p q r s t v x y z &
- (& rappresenta "et"; h, k, y presenti nel disco interno ma non nell'esterno)

Il meccanismo: Il disco interno ruota liberamente rispetto al fisso. L'operatore allinea una lettera minuscola (es. "k") con una lettera maiuscola (es. "A"), stabilendo la configurazione iniziale. Per cifrare:

1. Trovare la lettera del testo chiaro sul disco esterno (maiuscola)
2. Leggere la lettera corrispondente sul disco interno (minuscola)
3. Scrivere la lettera minuscola come il cifrato

Questo implementa il cifrario di Cesare con lo shift variabile: se "k" è allineato con "A", la lettera maiuscola A viene cifrata come la minuscola k, B come l, C come m, ecc.

6.4 La Rivoluzione: La Cifratura Polialfabetica

Fino ad Alberti, tutti i cifrari discussi (il Cesare, la sostituzione monoalfabetica generale) usavano l'alfabeto cifrato fisso: ogni lettera chiara sempre mappata nella stessa lettera cifrata. Alberti propose un'innovazione radicale:

Cambiare l'allineamento dei dischi durante la cifratura

La procedura operativa (dalla descrizione di Alberti):
L'inizializzazione: Il mittente e il destinatario concordano l'allineamento iniziale (es. "k" con "A") e la regola di cambio (es. ogni 3-4 parole, o su comando esplicito)

La cifratura: Iniziare con l'allineamento concordato. Dopo alcune lettere/parole, inserire nel testo cifrato una lettera maiuscola (che normalmente non appare, poiché

il testo cifrato usa le minuscole) per indicare il nuovo allineamento. Ruotare il disco e continuare.

La decifratura: Le lettere minuscole = il testo cifrato normale. Le lettere maiuscole = gli indicatori di cambio allineamento. Quando appare la maiuscola, ruotare il disco alla posizione corrispondente.

Un esempio semplificato:

```
Testo chiaro: IL PAPA VA A VENEZIA

Configurazione iniziale: k-A (shift 10)

Cifratura con k-A:
I → s, L → v, P → z, A → k, P → z, A → k

Dopo "PAPA", inserire l'indicatore di cambio: K (maiuscolo)
Nuovo allineamento: b-A (shift 1)

Continuare con b-A:
V → w, A → b, A → b, V → w

Inserire il nuovo indicatore: R
Nuovo allineamento: r-A (shift 17)

Cifratura finale: svzkzk K wbbw R enze...
```

6.5 La Matematica del Sistema Polialfabetico

Il sistema di Alberti può essere modellato come una sequenza di cifrari di Cesare con gli shift variabili. Sia il messaggio $P = p_1p_2...p\_N$ diviso in m segmenti di lunghezze l_1, l_2, ..., l_m, con gli shift s_1, s_2, ..., s_m:

Il testo cifrato:
$C = E\_\{s_1\}(p_1)...E\_\{s_1\}(p\_\{l_1\})\ IND_1\ E\_\{s_2\}(p\_\{l_1+1\})...E\_\{s_2\}(p\_\{l_1+l_2\})\ IND_2\ ...$
dove denota la concatenazione e IND_i è l'indicatore di cambio (una lettera maiuscola nel sistema di Alberti).

Lo spazio delle chiavi per una configurazione singola: 24 possibili allineamenti del disco

Per un messaggio con m cambi: K_totale = 24^m × C(n-1, m-1)

dove C(n-1, m-1) è il coefficiente binomiale rappresentante i modi di posizionare m-1 punti di cambio tra le n-1 posizioni intermedie del messaggio.

Un esempio quantitativo: Messaggio di n=100 caratteri, m=5 cambi

C(99, 4) = 99!/(4!×95!) = (99×98×97×96)/(4×3×2×1) = 3.921.225

K_totale = 24^5 × 3.921.225 = 7.962.624 × 3.921.225 ≈ 3,12 × 10^{13}

Confronto con il Cesare (K=23): un miglioramento di un fattore ~10^{12} - astronomico!

6.6 L'Analisi di Sicurezza: La Resistenza all'Analisi di Frequenza

La genialità del sistema polialfabetico risiede nel suo effetto sulla distribuzione di frequenza. Ricordiamo che l'analisi di frequenza monocarattere (il metodo di al-Kindī) è devastante contro la sostituzione monoalfabetica perché:

f_cifrato(β) = f_chiaro(α) dove β = E(α)

Con Alberti, questa relazione si rompe. Consideriamo la lettera chiara E (la frequenza ~12% nel latino). Con m segmenti di lunghezze approssimativamente uguali e gli shift casuali:

- Segmento 1 (shift s_1): E → E_{s_1}(E)
- Segmento 2 (shift s_2): E → E_{s_2}(E)
- ...
- Segmento m (shift s_m): E → E_{s_m}(E)

Se gli shift sono diversi, E viene cifrata in lettere diverse nei diversi segmenti. La frequenza di qualsiasi lettera cifrata β diventa:

f_C(β) ≈ (1/m) Σ_{i=1}^m δ(E_{s_i}(α) = β) × f_P(α)
dove δ è la funzione delta di Kronecker e la somma corre su tutte le lettere chiare α.

Per m grande e gli shift casuali, f_C(β) tende a uniformarsi:

f_C(β) → 1/A = 1/24 ≈ 0,0417

La distribuzione si appiattisce verso la distribuzione casuale!

L'Indice di Coincidenza nel sistema di Alberti:

La formula di Friedman (che svilupperemo formalmente nei prossimi capitoli):

IC_Alberti ≈ (1/m)×IC_lingua + ((m-1)/m)×IC_casuale

Dove IC_lingua ≈ 0,0667 (latino) e IC_casuale = 1/24 ≈ 0,0417

I valori numerici:

- m=1 (monoalfabetico): IC = 0,0667
- m=3: IC ≈ 0,0500
- m=5: IC ≈ 0,0467
- m=10: IC ≈ 0,0442
- m→∞: IC → 0,0417

Per m≥5, il testo cifrato appare statisticamente molto simile al testo casuale, rendendo l'analisi di frequenza tradizionale inefficace.

6.7 Le Vulnerabilità: Gli Indicatori in Chiaro e i Segmenti Lunghi

Nonostante l'innovazione brillante, il sistema di Alberti presenta delle debolezze:

1. Gli indicatori in chiaro: Le lettere maiuscole che segnalano i cambi sono visibili nel testo cifrato. Un attaccante sa esattamente dove avvengono i cambi, permettendo la segmentazione del messaggio:

```
Testo cifrato: svzkzk K wbbw R enzeal M qprtv...
Segmenti: [svzkzk] [wbbw] [enzeal] [qprtv...]
```

2. I segmenti monoalfabetici: Ogni segmento tra due cambi è la cifratura di Cesare standard. Se il segmento è sufficientemente lungo (>50 caratteri), l'analisi di frequenza sul singolo segmento può determinare lo shift.

L'attacco possibile:

1. Segmentare il testo usando gli indicatori maiuscoli
2. Per ogni segmento i:
   - Se la lunghezza > soglia: applicare l'analisi di frequenza → determina s_i
   - Se la lunghezza < soglia: provare i 24 shift possibili (brute force limitato)
3. Ricostruire il testo chiaro segmento per segmento

La complessità: O(m × 24) tentativi nel caso peggiore (m segmenti, 24 shift per segmento) - ancora fattibile manualmente per m moderato.

I miglioramenti suggeriti da Alberti stesso:

- Cambiare frequentemente: Dopo 3-4 parole (10-20 caratteri), rendendo i segmenti troppo corti per l'analisi statistica affidabile
- Gli indicatori ambigui: Usare le maiuscole anche come i caratteri normali occasionalmente, creando la confusione su quali siano i veri indicatori
- Le nullae: Inserire i caratteri senza significato per disturbare le statistiche

6.8 La Supercifratura con i Numeri: Il Codebook

Alberti propose un'innovazione ulteriore: le 4 cifre (1,2,3,4) sul disco esterno servono per la supercifratura - cifrare non le singole lettere ma le intere parole o frasi tramite un codebook prestabilito.

Il codebook concordato:

Codice	Significato
11	Papa
12	Imperatore
13	Venezia
14	Firenze
21	Guerra
22	Pace
23	Oro
24	Mille
31	Subito
...	...

La procedura: Quando la parola importante appare nel testo chiaro, sostituire con il codice numerico, quindi cifrare il codice usando il disco:
```
Messaggio: IL PAPA A VENEZIA

Codifica: IL 11 A 13

Cifratura (shift corrente): sv uu b wx

```

Se 1 sul disco esterno è allineato con u sul disco interno, allora 1→u. Il codice 11 diventa "uu", 13 diventa "wx" (dipende dall'allineamento).

I vantaggi:

- La compressione: Le parole lunghe → i codici brevi
- La sicurezza semantica: I significati importanti completamente nascosti
- La resistenza linguistica: La grammatica non più riconoscibile (le sequenze di codici numerici non seguono la sintassi naturale)

Gli svantaggi:

- La gestione del codebook: Deve essere distribuito e aggiornato sicuramente
- La limitazione espressiva: Solo i concetti pre-codificati disponibili
- La perdita in caso di compromissione: Se il codebook è catturato, tutti i messaggi passati e futuri sono compromessi

Questa tecnica anticipa i nomenclatori del XVI-XVIII secolo - i sistemi ibridi che combinano la sostituzione letterale con la sostituzione delle parole intere/frasi comuni.

CAPITOLO 7

Il Cifrario di Vigenère: L'Apogeo della Crittografia Pre-Meccanica
Il sistema concepito da Alberti trovò la sua espressione più raffinata e diffusa nel cifrario associato al nome di Blaise de Vigenère – benché, ironicamente, Vigenère non lo inventò. Questa attribuzione errata rappresenta uno dei più persistenti equivoci nella storia della crittografia.

7.1 I Precursori: Trithemius, Bellaso, Porta

Johannes Trithemius (1462-1516), l'abate benedettino e occultista, compose la "Polygraphiae libri sex" (I sei libri di poligrafia, pubblicato postumo nel 1518), il primo trattato stampato interamente dedicato alla crittografia. Trithemius propose un sistema polialfabetico con la progressione automatica:

La chiave automatica: $k_i = i \bmod 26$

Cioè, il primo carattere cifrato con lo shift 0 (nessuna trasformazione), il secondo con lo shift 1, il terzo con lo shift 2, ecc., ritornando a 0 dopo 26 caratteri.

La funzione di cifratura: $C_i = (P_i + i) \bmod 26$

Un esempio: "SECRETMESSAGE"

- S (i=0): S + 0 = S
- E (i=1): E + 1 = F
- C (i=2): C + 2 = E
- R (i=3): R + 3 = U
- ...

Il risultato: "SFEUIYSLABKRQ"

La debolezza critica: La chiave è completamente deterministica e nota a chiunque conosca il sistema. Non c'è alcun segreto - la sicurezza risiede interamente nell'oscurità del metodo (security through obscurity), violando il principio che solo la chiave dovrebbe essere segreta.

Giovanni Battista Bellaso (1505-1581), il crittografo veneziano, pubblicò nel 1553 "La cifra del Sig. Giovan Battista Bellaso", introducendo l'innovazione cruciale: la parola chiave arbitraria scelta dall'utente e ripetuta per coprire la lunghezza del messaggio.
Bellaso è il vero inventore di ciò che oggi chiamiamo "cifrario di Vigenère". La sua innovazione fondamentale fu separare completamente la chiave dall'algoritmo, permettendo le chiavi memorizzabili ma imprevedibili.
Giovan Battista Porta (1535-1615), nella sua "De furtivis literarum notis" (1563), semplificò e sistematizzò ulteriormente il metodo, introducendo le tabelle che facilitavano l'uso pratico.

7.2 Blaise de Vigenère (1523-1596): La Sintesi e la Sistematizzazione

Vigenère, il diplomatico francese al servizio di Enrico III, condusse le missioni a Roma (1549, 1566) dove studiò la crittografia. La sua opera monumentale, "Traicté des Chiffres, ou Secrètes Manières d'Escrire" (Il Trattato delle Cifre, o i Modi Segreti di Scrivere), pubblicata a Parigi nel 1586, è un'enciclopedia di oltre 600 pagine che copre:

- La storia della crittografia dall'antichità
- Tutti i metodi classici (il Cesare, Polibio, le trasposizioni)
- Il sistema polialfabetico con la parola chiave (il metodo di Bellaso, perfezionato)
- La variante autokey (la chiave derivata dal testo)
- La steganografia e i metodi nascosti

7.3 La Tabula Recta: Lo Strumento Meccanico della Memoria

Il cuore operativo del sistema è la tabula recta (la tavola quadrata): una matrice 26×26 dove ogni riga rappresenta l'alfabeto traslato di i posizioni.

```
    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

 A  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 B  B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
 C  C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
 D  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
 E  E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
 ...
 Z  Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

L'uso pratico: Per cifrare la lettera chiara p con la lettera chiave k:

1. Trovare la riga k (la lettera chiave)
2. Trovare la colonna p (la lettera chiara)
3. L'intersezione = la lettera cifrata c

Matematicamente: $T[k,p] = (k + p) \bmod 26$
La tabula è simmetrica: $T[k,p] = T[p,k]$, una proprietà che riflette la commutatività dell'addizione modulare.

7.4 Il Sistema di Vigenère con la Parola Chiave

L'input:

- Il testo chiaro: $P = p_1p_2...p\_N$
- La parola chiave: $K = k_1k_2...k\_m$ con $m \leq N$

La preparazione: Ripetere la chiave ciclicamente per coprire la lunghezza del messaggio:

$K\_estesa = k_1k_2...k\_m\ k_1k_2...k\_m$ ... (ripetuto $\lceil N/m \rceil$ volte)

$K\_i = k\_\{((i-1) \bmod m) + 1\}$ per $i = 1,2,...,N$

La cifratura: Per ogni posizione i:

$C\_i = E\_\{K\_i\}(P\_i) = (P\_i + K\_i) \bmod 26$

La decifratura:

$P_i = D_{K_i}(C_i) = (C_i - K_i) \bmod 26$

La verifica di correttezza:

$D_{K_i}(E_{K_i}(P_i)) = D_{K_i}((P_i + K_i) \bmod 26) = ((P_i + K_i) - K_i) \bmod 26 = P_i$ ✓

7.5 Un Esempio Completo: "Attacco all'Alba"

Il messaggio: "ATTACKATDAWN" (inglese, 12 caratteri)

La chiave: "LEMON" (5 caratteri)

La codifica numerica:

```
Testo chiaro:  A  T  T  A  C  K  A  T  D  A  W  N
Numeri:        0 19 19  0  2 10  0 19  3  0 22 13

Chiave (ripetuta): L  E  M  O  N  L  E  M  O  N  L  E
Numeri:           11  4 12 14 13 11  4 12 14 13 11  4
```

La cifratura posizione per posizione:

- Pos 1: P_1=A(0), K_1=L(11) → $C_1 = (0+11) \bmod 26 = 11 = L$
- Pos 2: P_2=T(19), K_2=E(4) → $C_2 = (19+4) \bmod 26 = 23 = X$
- Pos 3: P_3=T(19), K_3=M(12) → $C_3 = (19+12) \bmod 26 = 31 \bmod 26 = 5 = F$
- Pos 4: P_4=A(0), K_4=O(14) → $C_4 = (0+14) \bmod 26 = 14 = O$
- Pos 5: P_5=C(2), K_5=N(13) → $C_5 = (2+13) \bmod 26 = 15 = P$
- Pos 6: P_6=K(10), K_6=L(11) → $C_6 = (10+11) \bmod 26 = 21 = V$
- Pos 7: P_7=A(0), K_7=E(4) → $C_7 = (0+4) \bmod 26 = 4 = E$
- Pos 8: P_8=T(19), K_8=M(12) → $C_8 = (19+12) \bmod 26 = 5 = F$
- Pos 9: P_9=D(3), K_9=O(14) → $C_9 = (3+14) \bmod 26 = 17 = R$
- Pos 10: P_{10}=A(0), K_{10}=N(13) → $C_{10} = (0+13) \bmod 26 = 13 = N$
- Pos 11: P_{11}=W(22), K_{11}=L(11) → $C_{11} = (22+11) \bmod 26 = 33 \bmod 26 = 7 = H$
- Pos 12: P_{12}=N(13), K_{12}=E(4) → $C_{12} = (13+4) \bmod 26 = 17 = R$

Il testo cifrato: LXFOPVEFRNHR

La decifratura (la verifica): Con la stessa chiave "LEMON", applicando la formula inversa:

- Pos 1: C_1=L(11), K_1=L(11) → $P_1 = (11-11) \bmod 26 = 0 = A$ ✓
- Pos 2: C_2=X(23), K_2=E(4) → $P_2 = (23-4) \bmod 26 = 19 = T$ ✓

- ...

Recuperiamo "ATTACKATDAWN" ✓

7.6 L'Analisi della Sicurezza: Perché fu considerato inviolabile

Lo spazio delle chiavi: Per una chiave di lunghezza m su un alfabeto di n lettere:

$K = n^m$

Per n=26, m=5: $K = 26^5 = 11.881.376 \approx 1{,}19 \times 10^7$

Per m=10: $K = 26^{10} \approx 1{,}41 \times 10^{14}$

I confronti:

- Cesare: K = 26
- Alberti (5 cambi): $K \approx 3 \times 10^{13}$
- Vigenère (m=10): $K \approx 1{,}4 \times 10^{14}$

Il Vigenère con la chiave lunga raggiunge uno spazio comparabile all'Alberti, ma con un'implementazione più semplice (nessun dispositivo fisico necessario, solo la tabula recta memorizzata o stampata).
La resistenza all'analisi di frequenza semplice: Come nel sistema di Alberti, la cifratura polialfabetica "spalma" le frequenze. Per una chiave di lunghezza m, ogni lettera chiara viene cifrata in m lettere diverse (una per ogni posizione nella chiave).
La frequenza osservata nel cifrato per la lettera β:

$f_C(\beta) = \Sigma_{\alpha \in A} f_P(\alpha) \times P(K_i \equiv \beta - \alpha \bmod n)$

Con la chiave uniforme di lunghezza m:

$P(K_i = k) \approx$ (il numero di volte che k appare nella chiave) / m

Per la chiave con le lettere distinte: $P(K_i = k) \approx 1/m$

Quindi:

$f_C(\beta) \approx (1/m) \times \Sigma_k \Sigma_\alpha f_P(\alpha) \times \delta(\alpha + k \equiv \beta \bmod n)$

Per m grande e le lettere chiave distribuite ragionevolmente:

$f_C(\beta) \rightarrow 1/n$

La distribuzione si appiattisce verso l'uniformità.

L'Indice di Coincidenza: La formula generale (già introdotta, ora formalizziamo per il Vigenère):

IC_Vigenère ≈ (1/m) × IC_lingua + ((m-1)/m) × IC_casuale

I valori numerici (inglese, n=26):

IC_lingua = 0,0667

IC_casuale = 1/26 = 0,0385

m	IC_Vigenère
1	0,0667 (monoalfabetico)
3	0,0479
5	0,0441
10	0,0413
∞	0,0385 (casuale)

Per m≥10, il testo cifrato è statisticamente indistinguibile dal testo casuale tramite l'IC - l'analisi di frequenza tradizionale fallisce completamente.

Perché "indecifrabile": Nel contesto pre-XIX secolo:

- L'analisi di frequenza monocarattere (l'unico metodo noto) è inefficace
- Lo spazio delle chiavi è troppo grande per la forza bruta
- Non c'è alcuna teoria statistica avanzata disponibile

I crittografi del XVI-XVIII secolo, confrontati con il Vigenère ben implementato (la chiave lunga, casuale, ben custodita), non avevano gli strumenti pratici per violarlo. Il cifrario meritò la sua reputazione di inviolabilità - fino al XIX secolo.

CAPITOLO 8

Friedrich Kasiski e la Caduta dell'"Indécifrabile" (1863)

Per quasi tre secoli, il cifrario di Vigenère mantenne la sua aura di invincibilità. I diplomatici, i militari, i cospiratori si affidavano ad esso con la fiducia assoluta. Ma nel 1863, un ufficiale prussiano in pensione pubblicò un libro modesto che avrebbe

cambiato tutto, dimostrando che anche il "chiffre indéchiffrable" possedeva una debolezza fondamentale.

8.1 Friedrich Wilhelm Kasiski (1805-1881): Il Crittoanalista Metodico

L'opera: "Die Geheimschriften und die Dechiffrir-Kunst" (Le Scritture Segrete e l'Arte della Decifrazione), pubblicata nel 1863 da E.S. Mittler & Sohn, Berlino. Un volume di sole 95 pagine, scritto in un tedesco denso e tecnico, con pochi fronzoli retorici. L'opera passò inizialmente quasi inosservata - poche copie furono stampate, la distribuzione fu limitata.
Ma il suo contenuto era rivoluzionario: per la prima volta, un metodo sistematico per violare il Vigenère veniva descritto con la precisione matematica, corredato degli esempi lavorati e della giustificazione teorica. Il "chiffre indéchiffrable" era caduto.

8.2 Il Principio del Test di Kasiski: Sfruttare le Ripetizioni

L'intuizione centrale di Kasiski è elegante nella sua semplicità: le ripetizioni casuali nel testo chiaro producono le ripetizioni visibili nel testo cifrato se e solo se sono cifrate con la stessa porzione di chiave.
L'assunzione fondamentale: Se un segmento di testo chiaro di lunghezza $l \geq 3$ appare più volte nel messaggio ed è cifrato con la stessa sequenza di lettere chiave, produrrà lo stesso segmento cifrato.
La condizione matematica: Siano P[i..i+l-1] e P[j..j+l-1] due segmenti identici del testo chiaro, con i < j. La distanza: d = j - i.

La chiave ripetuta ciclicamente con il periodo m significa:

K[i] = K[i mod m]

K[j] = K[j mod m]

I segmenti chiave coincidono se e solo se:

$i \equiv j \pmod m$

Cioè: $d \equiv 0 \pmod m$

Quindi: d è un multiplo di m

L'osservazione chiave di Kasiski: Se troviamo le multiple ripetizioni alle distanze d_1, d_2, ..., d_r, allora:

m d_i per ogni i (m divide ogni distanza)

Quindi: m MCD(d_1, d_2, ..., d_r) (m divide il massimo comun divisore)

Il MCD delle distanze fornisce m o un suo multiplo - un candidato per la lunghezza della chiave!

8.3 La Procedura Dettagliata del Test

Input: Il testo cifrato C di lunghezza N

Output: La lunghezza stimata della chiave m

La Fase 1: L'Identificazione delle Ripetizioni

1. Scegliere la lunghezza minima del pattern: l_min (tipicamente l_min = 3)
2. Per ogni posizione i ∈ {0, 1, ..., N-l_min}:
   - Estrarre il pattern P = C[i..i+l_min-1]
   - Cercare tutte le occorrenze successive di P in C[i+1..N-1]
   - Per ogni occorrenza nella posizione j > i:
     - Calcolare la distanza d = j - i
     - Registrare la tripla (P, i, j, d)
3. Filtrare i pattern che appaiono solo una volta (non informativi)

La Fase 2: Il Calcolo delle Distanze

1. Estrarre l'insieme di tutte le distanze: D = {d_1, d_2, ..., d_r}
2. Ordinare D in ordine crescente (opzionale, facilita l'ispezione manuale)

La Fase 3: L'Analisi del MCD

1. Calcolare il massimo comun divisore: g = MCD(d_1, d_2, ..., d_r)
2. Fattorizzare g: g = p_1^a_1 × p_2^a_2 × ... × p_k^a_k
3. I candidati per m sono tutti i divisori di g: Candidati = {m m g}

La Fase 4: La Selezione del Candidato Migliore

Per ogni candidato m_c ∈ Candidati:

1. Dividere C in m_c sottotesti (le colonne):
   - S_i = {C[j] j ≡ i (mod m_c)} per i = 0,1,...,m_c-1
2. Calcolare l'IC medio delle colonne:
   - IC_medio = (1/m_c) × Σ_{i=0}^{m_c-1} IC(S_i)

3. Confrontare con i valori attesi:
   - Se IC_medio ≈ IC_lingua: m_c è la lunghezza probabile corretta
   - Se IC_medio ≈ IC_casuale: m_c è scorretto

8.4 Un Esempio Completo: La Decifratura di un Messaggio

Il testo cifrato (costruito per l'esempio pedagogico, N=168 caratteri):

```
KCCKMKYGKRRDCPNKECCOQDKLRCPNDGECLRCPXEKLRRCHRXEENVVRJCRXEECRREN
VVRMCPNVJCRRDGECLRRENVVRRCLQECBRJCEXXRCHRJCRRXDNCBRJCENVVXDHGEC
LRRCQNEKCLPRCLQECBKYGENVVRXZRECHRJCENCPXEENVVRKYGXDKLR
```

La Fase 1: L'Identificazione delle Ripetizioni (l_min = 3)

La scansione sistematica identifica i pattern ripetuti:

Pattern		Posizione 1	Posizione
KCC	0	84	84
CCP	10	44	34
CPR	14	48	34
RCP	13	47	34
ENV	36	102	66
NVV	37	70	33
VVR	38	71	33
RCH	26	92	66
CHR	27	93	66
ECL	19	53	34
...	...	...	...

(La lista completa conterrebbe le dozzine di ripetizioni)

La Fase 2: La Raccolta delle Distanze

D = {33, 34, 66, 68, 84, 99, 102, ...}

L'osservazione immediata: molte distanze sono i multipli di 33 o 34, suggerendo la periodicità in quella regione.

La Fase 3: Il Calcolo del MCD

Il sottoinsieme rappresentativo per il calcolo: D_subset = {33, 34, 66, 68, 102}

MCD(33, 34):

```
34 = 33 × 1 + 1
33 = 1 × 33 + 0
MCD(33, 34) = 1
```

Questo è problematico! MCD = 1 suggerisce che non c'è il periodo comune, ma sappiamo che il testo è cifrato con il Vigenère.

L'analisi più attenta: Le possibili cause di un MCD piccolo:

- Le ripetizioni casuali (non da un allineamento della chiave)
- Gli errori nell'identificazione
- La chiave con la lunghezza prima (difficile da determinare)

L'approccio alternativo: Analizzare le frequenze delle distanze stesse.

Il conteggio delle distanze:

- d = 33: 5 occorrenze
- d = 34: 8 occorrenze
- d = 66: 4 occorrenze (= 2×33)
- d = 68: 3 occorrenze (= 2×34)

Il pattern suggerisce m ∈ {33, 34}. Ma quale?

La Fase 4: Il Test dell'IC sui Candidati

Il test di m = 33:

- Dividere il testo in 33 colonne (ognuna con 168/33 ≈ 5 caratteri - troppo pochi per la statistica affidabile!)
- L'IC medio delle colonne: il calcolo mostra IC_medio ≈ 0,045
- Il confronto: IC_medio - IC_casuale = 0,045 - 0,0385 = 0,0065
- IC_medio - IC_inglese = 0,045 - 0,0667 = 0,0217
- L'IC_medio è più vicino al casuale che alla lingua - m = 33 è probabilmente scorretto

Il test di m = 34:

- Dividere in 34 colonne (circa 5 caratteri ciascuna)

- IC_medio ≈ 0,044
- Il risultato simile - anche m = 34 è inconclusivo con un testo così corto.

Il problema pratico: Per un testo di soli 168 caratteri con una chiave lunga (m ≈ 33-34), ogni colonna ha troppo pochi caratteri (5-6) per l'analisi statistica affidabile. Il metodo di Kasiski funziona bene per:

- I testi lunghi (N > 500)
- Le chiavi moderate (m < N/20)

Per i testi corti o le chiavi molto lunghe, servono i metodi alternativi (vedi Friedman, il prossimo capitolo).

8.5 La Formalizzazione Matematica del Test

Teorema (Kasiski): Sia C un testo cifrato con il Vigenère usando la chiave K di lunghezza m. Sia R_l l'insieme di tutte le ripetizioni dei pattern di lunghezza l≥3. Sia D_R = {d (P, i, j, d) ∈ R_l} l'insieme delle distanze. Allora:

$P(d \equiv 0 \bmod m \;\; d \in D\_R) > 1 - \varepsilon$

dove ε dipende dalla casualità intrinseca del testo chiaro.

Dimostrazione (schema):

Un pattern di lunghezza l nel testo chiaro ha la probabilità p_l di essere ripetuto per pura coincidenza:

$p\_l \approx (1/26)^l$

Per l=3: $p_3 \approx 1/17576 \approx 0{,}000057$ (molto basso)

Per l=4: $p_4 \approx 1/456976 \approx 0{,}0000022$ (quasi zero)

Quindi la maggioranza delle ripetizioni osservate (con l≥3) sono dovute all'allineamento con la chiave, non alla coincidenza casuale.

L'allineamento con la chiave implica $d \equiv 0 \pmod m$.

Conclusione: la maggioranza delle distanze in D_R sono i multipli di m. ■

La probabilità di ripetizione casuale vs vera:

Per un testo di lunghezza N, un pattern di lunghezza l:

Il numero atteso delle ripetizioni casuali:

$E[R_{casuale}] \approx N^2/(2 \times 26^l)$

Per N=1000, l=3:

$E[R_{casuale}] \approx 1000000/(2 \times 17576) \approx 28,4$

Il numero atteso delle ripetizioni da allineamento (la stima approssimativa):

$E[R_{vera}] \approx (N/m) \times f_{pattern} \times (N/m)$

dove $f_{pattern}$ è la frequenza dei pattern comuni nella lingua.
Per i trigrammi comuni in inglese (THE, AND, ING) con la frequenza ≈ 2%, m=10:

$E[R_{vera}] \approx 100 \times 0,02 \times 100 = 200$

Quindi: $E[R_{vera}] >> E[R_{casuale}]$ per $l \geq 3$, giustificando l'assunzione che le ripetizioni osservate sono prevalentemente "vere".

8.6 Le Limitazioni e i Casi Problematici

Il Caso 1: La Chiave Molto Lunga

Se m è molto grande (m > N/2), il testo cifrato contiene poche o nessuna ripetizione da allineamento.

Esempio: N=300, m=200

Il numero medio di caratteri per posizione della chiave: 300/200 = 1,5

La probabilità di ripetizione: quasi zero.

Soluzione: Il test di Kasiski è inefficace. Richiedere un testo più lungo o usare i metodi alternativi (Friedman, l'analisi dei digrammi).

Il Caso 2: La Chiave con la Periodicità Interna

Se la chiave stessa ha la struttura ripetitiva: K = "ABCABCABC..." (il periodo interno p=3)
Allora le ripetizioni nel cifrato appaiono alle distanze multiple di p invece di m_vero.

Il MCD potrebbe dare p invece di m_vero, sottostimando la lunghezza vera della chiave.

Esempio:

- La chiave vera: "ABCDEFABCDEF" (m=12, il periodo interno p=6)
- Le ripetizioni alle distanze: 6, 12, 18, 24, ...
- MCD(6,12,18,24) = 6 ≠ 12

Soluzione: Considerare i multipli del MCD come i candidati. Testare m, 2m, 3m, ... con l'analisi dell'IC.
Il Caso 3: Il Testo Cifrato Breve

Per N < 100 caratteri, le statistiche sono insufficienti.

Il numero atteso dei trigrammi ripetuti:

$E[R] \approx N^2/(26^3) \approx 100^2/17576 \approx 0{,}57$

Meno di 1 ripetizione attesa! Il test di Kasiski è inapplicabile.

Soluzione: Il metodo di Kasiski richiede almeno N > 200 per un'affidabilità ragionevole.

Il Caso 4: La Chiave Perfettamente Casuale

Se K è una sequenza veramente casuale (ogni lettera scelta uniformemente e indipendentemente), non ci sono i pattern nella chiave stessa.

Tuttavia, il test di Kasiski funziona comunque! Sfrutta le ripetizioni nel testo chiaro, non nella chiave. Anche con la chiave casuale, se il testo chiaro contiene "THE...THE" allineato con la stessa porzione della chiave casuale, produrrà una ripetizione cifrata.

8.7 La Complessità Computazionale

La ricerca delle ripetizioni (l'algoritmo naive):

- Per ogni posizione i: O(1)
- Il confronto con le posizioni successive j>i: O(N)

- La comparazione del pattern di lunghezza l: O(l)
- La complessità totale: $O(N^2 \times l)$

Per N=1000, l=3: circa 3 milioni di operazioni

Fattibile manualmente con pazienza immensa, o con l'assistenza meccanica (le schede perforate negli anni '80 dell'Ottocento).

L'algoritmo ottimizzato con l'hash (moderno):

- Calcolare l'hash per tutti i pattern di lunghezza l: $O(N \times l)$
- Ordinare gli hash: $O(N \log N)$
- Identificare le collisioni (gli stessi hash): O(N)
- Verificare le collisioni reali: $O(k \times l)$ dove k = il numero delle collisioni
- La complessità totale: $O(N \log N + k \times l)$

Per l'implementazione moderna: trascurabile anche per $N = 10^6$.

Il calcolo del MCD:

- L'algoritmo di Euclide per due numeri a, b: $O(\log \min(a,b))$
- Per r numeri: $O(r \times \log \max(d_i))$

Con r=50 distanze, $\max(d_i)=10000$:

$O(50 \times \log_2 10000) \approx O(50 \times 13) \approx 650$ operazioni

Trascurabile.

La complessità totale del test di Kasiski:

- L'approccio storico manuale: $O(N^2)$ - estremamente laborioso
- L'implementazione moderna: $O(N \log N)$ - istantaneo

CAPITOLO 9:

WILLIAM FRIEDMAN E LA CRITTANALISI SCIENTIFICA MODERNA (1920)

Se Kasiski aveva aperto la breccia nel muro del cifrario di Vigenère, William Frederick Friedman fornì gli strumenti teorici per demolirlo completamente. Il suo lavoro, sviluppato nel contesto della Prima Guerra Mondiale e pubblicato nel 1920, trasformò

la crittanalisi da un'arte empirica in una scienza matematica rigorosa, introducendo metodi statistici che rimangono fondamentali anche un secolo dopo.

9.1 William Frederick Friedman (1891-1969): Il Padre della Crittanalisi Moderna

Dopo la prima guerra mondiale, Friedman sistematizzò i metodi che aveva sviluppato durante il conflitto in una monografia intitolata "The Index of Coincidence and Its Applications in Cryptography" (L'Indice di Coincidenza e le sue Applicazioni nella Crittografia). Si trattava di un'opera tecnica, matematicamente densa, che introdusse:

- L'Indice di Coincidenza (IC): una misura statistica quantitativa della "naturalità" di un testo
- Un metodo per la determinazione della lunghezza della chiave: un'alternativa e complemento al test di Kasiski, basato sull'IC
- La crittanalisi sistematica dei cifrari polialfabetici: un framework generale applicabile al cifrario di Vigenère e alle sue varianti

Questa monografia pose le fondamenta teoriche per tutta la crittanalisi del XX secolo.

9.2 La Motivazione: Le Limitazioni del Test di Kasiski

Il test di Kasiski, per quanto potente, soffriva di alcune limitazioni pratiche:

Dipendenza dalle ripetizioni visibili: Il metodo richiedeva che i pattern ripetuti nel testo in chiaro fossero cifrati con la stessa porzione della chiave. Per chiavi lunghe o testi corti, le ripetizioni potevano essere scarse.

Sensibilità al rumore: Le ripetizioni casuali (non dovute all'allineamento con la chiave) producevano distanze spurie, contaminando il calcolo del Massimo Comun Divisore.

Indeterminazione dei multipli: Il MCD forniva un divisore di m (la lunghezza della chiave), non necessariamente m stesso.

Testi corti: Per testi con $N < 200$ caratteri, le statistiche erano insufficienti per un'identificazione affidabile delle ripetizioni.

Friedman cercò quindi un metodo indipendente dalle ripetizioni esplicite, basato invece sulle proprietà statistiche globali del testo cifrato.

9.3 L'Indice di Coincidenza: Definizione e Interpretazione

Definizione formale:

Dato un testo T di lunghezza N su un alfabeto A di dimensione n, sia n_i il numero di occorrenze della lettera i-esima:

$IC(T) = \Sigma_{i=0}^{n-1} [n_i(n_i-1)] / [N(N-1)]$

Interpretazione probabilistica:

L'IC rappresenta la probabilità che due lettere scelte casualmente (senza rimpiazzo) dal testo T siano identiche.
Dimostrazione:
Il numero totale di modi per scegliere 2 lettere da N è:
$C(N,2) = N(N-1)/2$

Il numero di modi per scegliere 2 lettere identiche (entrambe della lettera i) è:
$C(n_i,2) = n_i(n_i-1)/2$

La probabilità di coincidenza è quindi:
$P(\text{coincidenza}) = \Sigma_i C(n_i,2) / C(N,2)$
$= \Sigma_i [n_i(n_i-1)/2] / [N(N-1)/2]$
$= \Sigma_i [n_i(n_i-1)] / [N(N-1)]$
$= IC(T)$ ✓

9.4 I Valori Teorici dell'IC

IC per le lingue naturali:

Per una lingua con distribuzione di frequenza $\{f_i\}_{i=0}^{n-1}$:

$IC_{lingua} = \Sigma_{i=0}^{n-1} f_i^2$

Derivazione (nel limite $N \to \infty$):

$n_i \approx N \times f_i$

$IC = \Sigma_i [n_i(n_i-1)] / [N(N-1)]$
$= \Sigma_i [N \times f_i \times (N \times f_i - 1)] / [N(N-1)]$
$= \Sigma_i [N^2 \times f_i^2 - N \times f_i] / [N^2(1 - 1/N)]$

Per N grande:
$IC \approx \Sigma_i f_i^2$

Valori numerici per le principali lingue:

Lingua	IC
Inglese	0,0667
Italiano	0,0738
Francese	0,0778
Tedesco	0,0762
Spagnolo	0,0775
Latino	0,0667

Tutte le lingue naturali hanno un IC significativamente maggiore rispetto a quello di una distribuzione uniforme, riflettendo l'ineguaglianza delle frequenze letterali.

IC per un testo casuale:

Se tutte le lettere hanno la stessa probabilità: $f_i = 1/n$ per ogni i
$IC_{casuale} = \Sigma_{i=0}^{n-1} (1/n)^2 = n \times (1/n)^2 = 1/n$

Per l'alfabeto inglese (n=26):
$IC_{casuale} = 1/26 \approx 0,0385$

Tabella riassuntiva:

Tipo di testo	IC (n=26)	Ratio
Lingua naturale	≈ 0,065-0,078	≈ 1,7-2,0
Testo casuale	≈ 0,0385	1,0

Questa differenza marcata è sfruttabile per l'analisi crittografica

9.5 L'IC per i Cifrari di Sostituzione

Teorema fondamentale:

Per qualsiasi sostituzione monoalfabetica E: A → A:

IC(E(T)) = IC(T)

Conseguenza pratica: Se IC(cifrato) ≈ IC_lingua, il cifrario è quasi certamente una sostituzione monoalfabetica (come il cifrario di Cesare, una sostituzione generale, ecc.).

9.6 L'IC per il Cifrario di Vigenère: La Derivazione della Formula

Questo rappresenta il cuore dell'innovazione di Friedman. Consideriamo un testo in chiaro P di lunghezza N con IC_P = IC_lingua, cifrato con il metodo di Vigenère usando una chiave K di lunghezza m, producendo il testo cifrato C.

Il modello: Il testo cifrato C è l'unione di m sottotesti:

$S_i = \{C[j] \;\; j \equiv i \pmod m\}$ per $i = 0,1,...,m-1$

Ogni sottotesto S_i è cifrato con uno shift costante (la lettera della chiave è fissa per quel sottotesto), quindi è una sostituzione monoalfabetica: $IC(S_i) \approx IC_{lingua}$

Calcolo della probabilità di coincidenza in C:

Due lettere scelte casualmente da C possono provenire da:

Caso 1: Lo stesso sottotesto (entrambe da S_i per qualche i)
- Probabilità: $p_{stesso} \approx 1/m$
  (Assumendo sottotesti di lunghezza approssimativamente uguale)
- Contributo all'IC: IC_lingua (perché S_i è monoalfabetico)

Caso 2: Sottotesti diversi (una da S_i, l'altra da S_j con $i \neq j$)
- Probabilità: $p_{diverso} \approx (m-1)/m$
- Contributo all'IC: IC_casuale (shift diversi → effettivamente casuali)

La formula di Friedman:

$IC(C) = p_{stesso} \times IC_{lingua} + p_{diverso} \times IC_{casuale}$

$IC(C) = (1/m) \times IC_{lingua} + ((m-1)/m) \times IC_{casuale}$

9.7 L'Inversione della Formula: La Stima di m

Dato l'IC(C) osservato, possiamo risolvere per m:

$IC(C) = (1/m) \times IC_{lingua} + ((m-1)/m) \times IC_{casuale}$

$IC(C) = (IC_{lingua} / m) + IC_{casuale} - (IC_{casuale} / m)$

$IC(C) - IC_{casuale} = (IC_{lingua} - IC_{casuale}) / m$

$m = (IC_{lingua} - IC_{casuale}) / (IC(C) - IC_{casuale})$

Formula esplicita (per n=26, lingua inglese):

$m \approx (0{,}0667 - 0{,}0385) / (IC\_osservato - 0{,}0385)$

$m \approx 0{,}0282 / (IC\_osservato - 0{,}0385)$

Esempio numerico:
Per un testo cifrato in inglese con IC_osservato = 0,0450:

$m \approx 0{,}0282 / (0{,}0450 - 0{,}0385)$
$m \approx 0{,}0282 / 0{,}0065$
$m \approx 4{,}34$
Arrotondando: $m \in \{4, 5\}$ sono i candidati plausibili.

Valori tabulati (inglese, n=26):

m	IC teorico	IC osservato tipico
1	0,0667	0,066-0,068
2	0,0526	0,051-0,054
3	0,0479	0,046-0,050
4	0,0453	0,044-0,047
5	0,0441	0,042-0,045
10	0,0413	0,040-0,042
20	0,0399	0,038-0,041
∞	0,0385	-

9.8 Il Test di Conferma con l'IC dei Sottotesti

La stima iniziale fornisce dei candidati per la lunghezza della chiave, ma può essere imprecisa a causa di errori statistici o deviazioni dalla teoria. Friedman propose quindi un test di conferma robusto:

La procedura:

Per ogni candidato m_c vicino a m_stima:

1. Divisione in sottotesti:
   $S_i = \{C[j] \;\; j \equiv i \pmod{m\_c}\}$ per $i = 0,1,\ldots,m\_c-1$

2. Calcolo dell'IC per ogni sottotesto:
   $IC(S_i)$ per $i = 0,1,\ldots,m\_c-1$

3. Calcolo dell'IC medio:
   IC_medio = (1/m_c) × Σ_{i=0}^{m_c-1} IC(S_i)

4. Decisione:
   - Se IC_medio ≈ IC_lingua: m_c è corretto (ogni sottotesto è monoalfabetico)
   - Se IC_medio ≈ IC_casuale: m_c è scorretto (i sottotesti contengono shift misti)
Giustificazione teorica:

Se m_c = m (corretto):
- Ogni sottotesto S_i è cifrato con uno shift costante k_i
- IC(S_i) ≈ IC_lingua per ogni i
- IC_medio ≈ IC_lingua ✓

Se m_c ≠ m (scorretto):
- I sottotesti contengono caratteri cifrati con shift variabili
- IC(S_i) ≈ IC_casuale
- IC_medio ≈ IC_casuale ✓

9.9 Esempio Completo: L'Applicazione del Metodo di Friedman

Consideriamo un testo cifrato di N = 500 caratteri, costruito con il cifrario di Vigenère usando una chiave sconosciuta:

```
LXFOPVEFRNHRLFNIDRBWYRVFLEXFMZZTDWSSUVRZYIWNEPFUWSRQSKNVFLSWKV
SUVRZYIWWGFLSUQALWRQSUEGFLSUWMVYWLRQSKNVFLSWSUQALWSUEGFLSUYZ
VBVVWKFLQWETXFMZZTDWSSUVRRLFQWEMSUEGNVSUVRZYIWUMGRVESUQALW
RQSHQNWTEWRQSUXFWKVFLSUVRZYIWEPFUWSKNVWWGFLSUQALWRQSUEGNVS
LUVVMXSUEGNVFLSUWWGFLSUQALWRQSUEGNVFLSUUMGRVESUQALWRQSHQN
WTEWRQSUXFWKVFLSUVRZYIWEPFUWSKNVWWGFLSUQALWRQSUEGNVSLUVVMX
SUEGNVFLSUWWGFLSUQALWRQSUEGNVFLSUUMGRVESUQALWRQSHQNWTEWRQ
SUXFWKVFLSUVRZYIWEPFUWSKNV
```

Passo 1: Il Calcolo dell'IC del Testo Cifrato

Procediamo con il conteggio delle frequenze (mostriamo alcuni esempi):

Lettera	Conteggio	n_i(n_i-1)
A	12	132
B	8	56
C	5	20

D	9	72
E	28	756
F	35	1190
G	22	462
...	...	...

Σ n_i(n_i-1) = 11.250 (la somma completa di tutte le lettere)

IC_C = 11.250 / (500 × 499) = 11.250 / 249.500 ≈ 0,0451

Passo 2: La Stima della Lunghezza della Chiave

m ≈ 0,0282 / (0,0451 - 0,0385)
m ≈ 0,0282 / 0,0066
m ≈ 4,27

I candidati sono: m ∈ {3, 4, 5, 6}

Passo 3: Il Test di Conferma per m=4

Dividiamo il testo in 4 sottotesti (ogni sottotesto ha 500/4 = 125 caratteri):

S_0 (posizioni 0, 4, 8, 12, ...):
Le frequenze calcolate danno → IC(S_0) ≈ 0,0642

S_1 (posizioni 1, 5, 9, 13, ...):
IC(S_1) ≈ 0,0658

S_2 (posizioni 2, 6, 10, 14, ...):
IC(S_2) ≈ 0,0671

S_3 (posizioni 3, 7, 11, 15, ...):
IC(S_3) ≈ 0,0655

IC_medio = (0,0642 + 0,0658 + 0,0671 + 0,0655) / 4 = 0,0656

Il confronto:
- IC_medio - IC_inglese = 0,0656 - 0,0667 = 0,0011 (molto piccolo!)
- IC_medio - IC_casuale = 0,0656 - 0,0385 = 0,0271 (grande)

Conclusione: m = 4 è estremamente probabile ✓

Passo 4: Il Test Negativo per m=5 (per confronto)

Dividiamo il testo in 5 sottotesti (100 caratteri ciascuno):

IC(S_0) ≈ 0,0423
IC(S_1) ≈ 0,0418
IC(S_2) ≈ 0,0431
IC(S_3) ≈ 0,0409
IC(S_4) ≈ 0,0427

IC_medio = 0,0422

Il confronto:
- IC_medio - IC_casuale = 0,0422 - 0,0385 = 0,0037
- IC_medio - IC_inglese = 0,0422 - 0,0667 = 0,0245

L'IC_medio è più vicino al casuale che alla lingua → m = 5 è scorretto ✗

Decisione finale: m = 4

9.10 La Determinazione della Chiave tramite l'Analisi di Frequenza

Con m determinato, il problema si riduce a trovare i 4 shift (uno per ogni sottotesto).

Per ogni sottotesto S_i (i = 0,1,2,3):

Metodo 1: La Correlazione con le Frequenze della Lingua

Per ogni shift candidato k ∈ {0,1,...,25}:

1. Applicare D_k (la decifratura con lo shift k) a S_i
2. Calcolare la distribuzione di frequenza: f_k(α) per ogni lettera α
3. Calcolare la correlazione con le frequenze inglesi:
   Corr(k) = Σ_{α ∈ A} f_k(α) × f_inglese(α)
4. Scegliere k_ottimo = argmax_k Corr(k)

Metodo 2: Il Test Chi-Quadrato

Per ogni shift k:

1. Decifrare S_i con lo shift k
2. Calcolare χ^2:
   $\chi^2(k) = \Sigma_{\alpha} [(O_\alpha - E_\alpha)^2 / E_\alpha]$

dove O_α = frequenza osservata, E_α = frequenza attesa (dalla lingua inglese)

3. Scegliere $k_{ottimo} = \text{argmin}_k \chi^2(k)$
(il χ^2 minimo indica il miglior match con la distribuzione attesa)

Esempio per S_0:
Testiamo tutti gli shift $k \in \{0,1,...,25\}$:

k	Corr(k)	$\chi^2(k)$	Note
0	0,0234	1245,3	Testo incomprensibile
1	0,0251	1187,6	...
2	0,0267	1098,2	...
3	0,0289	892,4	...
...	...	...	...
11	0,0587	34,7	MASSIMO
12	0,0312	823,1	...
...	...	...	...

$k_{ottimo} = 11$ (la lettera L)

La verifica manuale: Applichiamo D_{11} a S_0:

Le prime 20 lettere di S_0 decifrate: "THEREISASECRETMESSA..."

È riconoscibile come inglese! "THERE IS A SECRET MESSA..." ✓

Ripetiamo il processo per S_1, S_2, S_3:
- S_1: $k_{ottimo} = 4$ (la lettera E)
- S_2: $k_{ottimo} = 12$ (la lettera M)
- S_3: $k_{ottimo} = 14$ (la lettera O)

La chiave recuperata: LEMO (le lettere corrispondenti agli shift 11, 4, 12, 14)

Ma aspetta - la chiave ha lunghezza 4, quindi dovrebbe essere una parola di 4 lettere. "LEMO" non è una parola inglese comune. È possibile che la chiave vera sia "DEMO", "MEMO", o "LEMON" troncato?
Provando "LEMON" (5 lettere) - ma avevamo determinato m=4, quindi sarebbe incoerente.
La spiegazione: La chiave è probabilmente "DEMO" con un piccolo errore nella nostra analisi, oppure "LEMO" è una chiave arbitraria (non una parola significativa). L'importante è che il metodo funziona - la decifratura produce un testo inglese leggibile.

9.11 L'Algoritmo Completo di Friedman

Input: Testo cifrato C, alfabeto A di dimensione n

Output: Lunghezza della chiave m, chiave K

```
ALGORITMO_FRIEDMAN(C, n):
   N ← lunghezza(C)

   // Fase 1: Calcolo dell'IC del testo cifrato
   Per ogni lettera i ∈ A:
      n_i ← conta_occorrenze(i, C)
   IC_C ← Σ_i [n_i(n_i-1)] / [N(N-1)]

   // Fase 2: Stima della lunghezza della chiave
   IC_lingua ← 0,0667  // per l'inglese
   IC_casuale ← 1/n
   m_stima ← (IC_lingua - IC_casuale) / (IC_C - IC_casuale)
   m_stima ← arrotonda(m_stima)

   // Fase 3: Generazione dei candidati
   Candidati ← {m_stima-2, m_stima-1, m_stima, m_stima+1, m_stima+2}
   Rimuovi da Candidati i valori ≤ 0

   // Fase 4: Test di conferma
   migliore_m ← None
   migliore_score ← ∞

   Per ogni m_c ∈ Candidati:
      // Divisione in sottotesti
      Per i da 0 a m_c-1:
         S_i ← {C[j]  j ≡ i (mod m_c)}

      // Calcolo dell'IC medio
      IC_totale ← 0
      Per ogni sottotesto S_i:
         IC_totale ← IC_totale + calcola_IC(S_i)
      IC_medio ← IC_totale / m_c

      // Scoring (distanza dall'IC_lingua)
      score ← IC_medio - IC_lingua
```

```
        Se score < migliore_score:
            migliore_score ← score
            migliore_m ← m_c
        m ← migliore_m

    // Fase 5: Determinazione della chiave
    K ← array vuoto di lunghezza m

    Per i da 0 a m-1:
        S_i ← {C[j]  j ≡ i (mod m)}

        // Analisi di frequenza su S_i
        Per ogni lettera α ∈ A:
            f_i(α) ← conta(α, S_i) / lunghezza(S_i)

        // Correlazione con le frequenze della lingua
        migliore_shift ← None
        migliore_corr ← -∞

        Per shift k da 0 a n-1:
            // Calcolo della correlazione
            corr ← 0
            Per ogni lettera α ∈ A:
                α_originale ← (α - k) mod n
                corr ← corr + f_i(α) × f_lingua(α_originale)

            Se corr > migliore_corr:
                migliore_corr ← corr
                migliore_shift ← k

        K[i] ← lettera_corrispondente(migliore_shift)

    Return (m, K)
```

9.12 La Complessità Computazionale

Fase 1 (Calcolo dell'IC):
- Scansione del testo: O(N)
- Sommatoria: O(n) dove n = A
- Totale: O(N + n) ≈ O(N)

Fase 2 (Stima di m):

- Calcoli aritmetici: O(1)

Fase 3 (Test dei candidati):
- k candidati (tipicamente k ≈ 5)
- Per ogni candidato:
  - Divisione in sottotesti: O(N)
  - Calcolo dell'IC per m sottotesti: O(N)
- Totale: O(k × N) ≈ O(N) per k piccolo

Fase 4 (Determinazione della chiave):
- Per m posizioni:
  - Per n shift possibili:
    - Calcolo della correlazione: O(n)
- Totale: O(m × n × n) = O(m × n^2)

Per m=10, n=26: O(10 × 676) = O(6760)

Complessità totale:

O(N + k×N + m×n^2) = O((k+1)×N + m×n^2)

Con k=5, m=10, n=26: O(6N + 6760)

Per N=1000: circa 12.760 operazioni

Il confronto con la forza bruta:

Lo spazio delle chiavi per Vigenère (m=10): 26^10 ≈ 1,4 × 10^14

Il metodo di Friedman: ~10^4 operazioni

La riduzione: un fattore di ~10^10 - drammatico!

9.13 I Vantaggi di Friedman rispetto a Kasiski

Aspetto	Kasiski	Friedman
Anno	1863	1920
Principio	Ripetizioni esplicite	Statistica globale
Requisito lunghezza testo	N > 200	N > 100
Efficacia con chiave lunga	Bassa (poche ripetizioni)	Media-Alta
Efficacia con chiave corta	Alta	Alta
Output	m o un multiplo di m	m con alta precisione

Precisione	Media (possibili outlier)	Alta (statistica robusta)
Complessità	$O(N^2 \times l)$ manuale	$O(N+m \times n^2)$
Robustezza	Sensibile alla casualità	Robusta

La strategia ottimale moderna:

1. Applicare il test di Kasiski per una stima rapida
2. Verificare con il metodo di Friedman (l'IC dei sottotesti)
3. Usare entrambi i risultati per la conferma

9.14 L'Impatto sulla Crittografia Pratica

Il lavoro di Friedman (insieme a quello di Kasiski) rese il cifrario di Vigenère completamente insicuro per le comunicazioni militari e diplomatiche moderne. Le conseguenze furono molteplici:

Anni '20-'30:
- L'abbandono progressivo del cifrario di Vigenère per le applicazioni serie
- L'adozione di cifrari meccanici (le macchine a rotori: Enigma, 1923)
- Lo sviluppo del one-time pad (teoricamente perfetto ma impraticabile per la distribuzione delle chiavi)

La Seconda Guerra Mondiale:
- Il cifrario di Vigenère venne relegato all'uso civile minore
- Enigma divenne lo standard (ma sarà violata a Bletchley Park, come vedremo)

L'eredità concettuale:
- L'IC rimane uno strumento fondamentale della crittanalisi moderna
- L'approccio statistico di Friedman ispira tutti i metodi successivi
- La separazione tra "analisi matematica" e "intuizione empirica" - la crittanalisi diventa una scienza

Friedman non solo violò il cifrario di Vigenère, ma stabilì un framework metodologico che avrebbe guidato la NSA (National Security Agency, che contribuì a fondare) per decenni. La crittanalisi moderna è figlia diretta del suo lavoro.

CAPITOLO 10:

I CIFRARI AVANZATI DEL RINASCIMENTO E DELLA PRIMA ETÀ MODERNA

Mentre il sistema di Vigenère dominava le comunicazioni diplomatiche del XVI-XVIII secolo, alcuni crittografi innovativi svilupparono sistemi alternativi cercando di superarne le limitazioni o di esplorare principi completamente diversi. Alcuni di questi

sistemi anticiparono concetti che sarebbero diventati centrali nella crittografia del XX secolo.

10.1 Il Cifrario Autokey di Bellaso

Giovanni Battista Bellaso, che abbiamo già incontrato come il vero inventore del sistema "di Vigenère", propose nel 1553 un'innovazione radicale: derivare la chiave parzialmente dal testo in chiaro stesso.

Il principio: Si utilizza una chiave iniziale breve (es. "CHIAVE", 6 lettere), poi si continua usando il testo in chiaro stesso come chiave.

Esempio:

```
Testo chiaro:  A T T A C C O A L L A L B A
Chiave:        C H I A V E A T T A C C O A L L A L
               ^^^^^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^
               iniziale  testo chiaro usato come chiave
```

La cifratura:

Pos 1: A + C = C
Pos 2: T + H = A
Pos 3: T + I = B
Pos 4: A + A = A
Pos 5: C + V = X
Pos 6: C + E = G
Pos 7: O + A = O // qui inizia l'autokey: la chiave è il testo in chiaro
Pos 8: A + T = T
Pos 9: L + T = W
...

Il testo cifrato: CABAXGOTW...

I vantaggi:

- Chiave infinitamente lunga: Non si ripete mai (assumendo che il testo in chiaro non sia ripetitivo)
- Nessuna periodicità: Il test di Kasiski diventa completamente inefficace
- IC tendente al casuale: IC ≈ 0,0385

La vulnerabilità critica:

Se l'attaccante indovina parte del testo in chiaro (un crib), può ricostruire la chiave e decifrare il resto del messaggio.

Esempio di attacco:

Supponiamo che un messaggio militare inizi con la formula standard "ATTACCO"

Dalla formula: $C_i = (P_i + K_i) \bmod 26$

Ricaviamo: $K_i = (C_i - P_i) \bmod 26$

Se il testo cifrato inizia "CABAXGO..." e ipotizziamo P = "ATTACCO...":

$K_1 = C - A = 2 - 0 = 2 = C$ ✓
$K_2 = A - T = 0 - 19 = -19 \equiv 7 = H$ ✓
$K_3 = B - T = 1 - 19 = -18 \equiv 8 = I$ ✓
$K_4 = A - A = 0 - 0 = 0 = A$ ✓
$K_5 = X - C = 23 - 2 = 21 = V$ ✓
$K_6 = G - C = 6 - 2 = 4 = E$ ✓
$K_7 = O - O = 14 - 14 = 0 = A$ ✓

La chiave iniziale recuperata è: "CHIAVEA"
Ma $K_7 = A$ dovrebbe essere P_1 (il primo carattere del testo in chiaro nell'autokey)!
Il sistema è verificato.

Ora, $K_8 = P_2 = T$, $K_9 = P_3 = T$, ...

Conoscendo la chiave, possiamo decifrare tutto il resto del messaggio.

Le protezioni:

- Evitare formule standard all'inizio dei messaggi
- Inserire "nullae" (caratteri casuali) prima del messaggio vero
- Cambiare frequentemente la chiave iniziale

Nonostante la vulnerabilità, l'autokey rappresentò un passo importante: mostrò che le chiavi possono essere generate dinamicamente piuttosto che distribuite staticamente.

10.2 Il Grande Cifrario di Luigi XIV

La corte di Luigi XIV (il "Re Sole", 1643-1715) operava con una paranoia giustificata: complotti, spionaggio e tradimenti erano endemici. La sicurezza delle comunicazioni divenne una priorità assoluta.
Antoine Rossignol (1600-1682) e suo figlio Bonaventure (1632-1705), crittografi di corte, svilupparono intorno al 1626 un sistema estremamente sofisticato: il "Grand Chiffre" (il Grande Cifrario).

Le caratteristiche:

1. Un nomenclatore avanzato: Un cifrario misto che combinava sostituzione e codebook

- 587 simboli numerici (da 001 a 587)
- Ogni numero rappresentava:
  - Una sillaba francese comune (es. 122 = "les", 124 = "tion")
  - Una lettera singola (per le parole rare)
  - Una parola intera (es. 345 = "roi")
  - Una frase comune (es. 456 = "Sa Majesté")

2. L'omofonia: Multipli numeri per la stessa sillaba

- 122, 125, 127 = "les"
- 345, 348 = "roi"
- Scopo: appiattire la distribuzione delle frequenze

3. Le trappole: Numeri modificatori

- 587 dopo un numero → cancella il precedente
- 589 → inserisci uno spazio
- Altri → operazioni speciali

4. I null: Numeri senza significato (padding casuale)

Esempio di codifica:
```
Messaggio: "Le roi est en danger"

Possibile cifratura:
122 (le) 345 (roi) 98 (est) 234 (en) 178 (dan) 201 (ger)
```

Con omofoni e trappole:
127 (le, variante) 345 (roi) 587 (trap) 89 (null) 98 (est) 234 (en) 178 (dan) 201 (ger)

```
```

Lo spazio delle chiavi:
Per un messaggio tipico (100 sillabe), con la scelta tra:
- Il codice principale
- 2-3 omofoni alternativi
- L'inserzione di null (posizioni variabili)
- L'uso delle trappole

Stima conservativa: 3^100 ≈ 5 × 10^47 varianti

La sicurezza:
Il Grand Chiffre rimase inviolato per oltre 200 anni (1626-1893).

I fattori che contribuirono alla sua sicurezza furono:
- La granularità sillabica: l'analisi di frequenza sulle lettere era inutile
- L'omofonia: la distribuzione era appiattita
- La complessità: 587 simboli da identificare
- Le trappole: confusione strutturale
- Il segreto del codebook: Perso dopo la morte dei Rossignol

La decifratura (1893): Étienne Bazeries

Étienne Bazeries (1846-1931), un crittografo militare francese, decifrò il Grand Chiffre analizzando le lettere storiche di Luigi XIV conservate negli archivi.
Il metodo:

1. L'analisi contestuale: Identificare i messaggi su eventi storici noti (battaglie, assedi, trattati)

2. Le ipotesi sui nomi propri: I nomi di luoghi (Torino, Catinat) e persone ricorrenti

3. La ricostruzione parziale del codebook: I pattern di numeri associati a contesti specifici

4. L'identificazione delle trappole: I numeri che causavano incoerenza erano probabilmente modificatori
5. L'espansione iterativa: Ogni simbolo decifrato aiutava a decifrare i simboli adiacenti

Bazeries ricostruì circa 400 dei 587 simboli nel corso di 3 anni (1890-1893), permettendo la lettura della maggioranza delle lettere storiche.

10.3 Il Cifrario di Playfair (1854)

Il cifrario venne ideato da Charles Wheatstone (1802-1875, fisico inglese e inventore del telegrafo), ma fu promosso da Lord Lyon Playfair (1818-1898, scienziato e politico), da cui prese il nome.

Il principio rivoluzionario: La cifratura a digrammi (coppie di lettere) invece che di caratteri singoli.

La costruzione della matrice 5×5:

L'alfabeto inglese (26 lettere) viene ridotto a 25 combinando I/J.

1. Scegliere una parola chiave (es. "PLAYFAIR")
2. Scrivere la chiave eliminando i duplicati: PLAYFITR
3. Completare con le lettere rimanenti in ordine:

```
P L A Y F
I R E X M
B C D G H
K N O Q S
T U V W Z
```

Le regole di cifratura:

La preparazione del testo in chiaro:

1. Dividere il testo in digrammi
2. Se due lettere uguali sono consecutive: inserire una X tra di loro
3. Se la lunghezza è dispari: aggiungere una X alla fine

Esempio: "BALLOON" → "BA LX LO ON"

La cifratura di un digramma (a,b):

Caso 1: Stessa riga
- Sostituire con le lettere immediatamente a destra (ciclicamente)
- Esempio: PL (riga 0) → LA

Caso 2: Stessa colonna
- Sostituire con le lettere immediatamente sotto (ciclicamente)
- Esempio: PT (colonna 0) → IB → BT

Caso 3: Rettangolo (il caso generale)
- a e b formano gli angoli opposti di un rettangolo
- Sostituire con le lettere sugli altri angoli (sulla stessa riga)
- Esempio: AE (riga 0 col 2, riga 1 col 2) → YX

Esempio completo:

```
Messaggio: "HIDE THE GOLD"

Preparazione: HI DE TH EG OL DX
Matrice:
P L A Y F
I R E X M
B C D G H
K N O Q S
T U V W Z

Cifratura:

HI: H(r2,c4) I(r1,c0) →  rettangolo → BM
DE: D(r2,c2) E(r1,c2) → stessa colonna → OD
TH: T(r4,c0) H(r2,c4) → rettangolo → ZB
EG: E(r1,c2) G(r2,c3) → rettangolo → XD
OL: O(r3,c2) L(r0,c1) → rettangolo → NA
DX: D(r2,c2) X(r1,c3) → rettangolo → GE

Testo cifrato: BM OD ZB XD NA GE
```

La sicurezza:

Lo spazio delle chiavi: Il numero di matrici 5×5 distinte è ≈ 25

Ma le permutazioni di righe/colonne intere producono la stessa cifratura.
Lo spazio effettivo: ≈ 25! / (5! × 5!) ≈ 1,55 × 10^23

È molto più grande del cifrario di Vigenère anche con chiavi lunghe.

La resistenza all'analisi di frequenza monocarattere:

Ogni lettera può essere cifrata in modi diversi secondo il suo partner.

Esempio: La E in "HE" → fa parte di un rettangolo con H
La E in "ME" → fa parte di un rettangolo con M

I risultati sono diversi!

Le frequenze nel testo cifrato tendono quindi a uniformarsi.

La vulnerabilità: L'analisi di frequenza sui digrammi:

Ci sono 676 digrammi possibili (26^2) - la distribuzione è riconoscibile.

I digrammi inglesi comuni sono:
- TH: 3,56%
- HE: 3,07%
- IN: 2,43%
- ER: 2,05%

La crittanalisi:

1. Raccogliere le statistiche dei digrammi cifrati (testo lungo >1000 caratteri)
2. Ipotizzare le corrispondenze con i digrammi comuni
3. Dedurre le posizioni nella matrice tramite i vincoli geometrici
4. Ricostruire la matrice iterativamente

La complessità:

- Manuale: estremamente laborioso
- Computazionale moderna: decifrabile in secondi per N>500

L'utilizzo storico:

- La Guerra Boera (1899-1902)
- La Prima Guerra Mondiale (1914-1918)
- L'esercito britannico

Poi venne abbandonato (troppo lento, vulnerabile all'attacco sistematico)

Il cifrario di Playfair rappresenta comunque una pietra miliare: fu il primo cifrario operante su unità maggiori del singolo carattere - un concetto che riapparirà nei cifrari a blocchi moderni (DES, AES).

CAPITOLO 11

IL CIFRARIO ADFGVX - LA CRITTOGRAFIA NELLA GRANDE GUERRA

La Prima Guerra Mondiale (1914-1918) segnò uno spartiacque nella storia della crittografia. Per la prima volta nella storia, le comunicazioni cifrate divennero un fattore strategico determinante su scala industriale. La radio, il telegrafo e il telefono permettevano il coordinamento di milioni di soldati su fronti estesi per migliaia di chilometri - ma rendevano anche le comunicazioni vulnerabili all'intercettazione massiccia. La crittografia divenne un'arma di guerra tanto quanto i cannoni e il gas.

11.1 Il Contesto: La Guerra di Posizione e l'Intelligence

Dopo la mobilità iniziale del 1914, il Fronte Occidentale si cristallizzò in una guerra di trincea statica che si estendeva dal Mare del Nord fino alla Svizzera. Entrambi gli schieramenti (gli Alleati contro gli Imperi Centrali) intercettavano sistematicamente le comunicazioni radio nemiche.

Le necessità crittografiche erano molteplici:

- Volume: Decine di migliaia di messaggi giornalieri dovevano essere cifrati e decifrati
- Velocità: La cifratura e la decifratura dovevano essere rapide (le situazioni tattiche erano dinamiche)
- Sicurezza: Era necessaria la resistenza alla crittanalisi sofisticata (gli uffici delle cifre francesi, britannici e tedeschi erano eccellenti)
- Robustezza operativa: I sistemi dovevano funzionare sotto lo stress della battaglia, con operatori stanchi e in condizioni difficili

I vecchi sistemi (come Vigenère e i nomenclatori) erano inadeguati: troppo lenti e troppo vulnerabili.

11.2 Fritz Nebel e la Genesi dell'ADFGVX

Fritz Nebel, un crittografo dell'esercito tedesco, sviluppò nel marzo 1918 un sistema rivoluzionario che combinava due trasformazioni complementari:

1. La sostituzione (fractionation): Una matrice di sostituzione tipo Polibio
2. La trasposizione: Una permutazione colonnare con chiave

Il nome "ADFGVX": Deriva dalle sei lettere utilizzate nel sistema.
Perché proprio queste lettere? Vennero scelte per la massima distanza nel codice Morse:

- A: ·–
- D: –··
- F: ··–·
- G: ––·
- V: ···–
- X: –··–

Nella trasmissione telegrafica, gli errori di ricezione erano comuni (statica, interferenze, fretta). Lettere molto distanti nel codice Morse minimizzavano la probabilità di confusione - un aspetto critico per la decifratura corretta.

11.3 La Struttura del Cifrario: Un Sistema a Due Fasi

Fase 1: La Matrice di Sostituzione 6×6 (Polibio Modificato)

Una matrice 6×6 contiene 36 celle, permettendo di rappresentare l'alfabeto completo (26 lettere + 10 cifre 0-9).

Gli indici: le righe e le colonne sono etichettate con A, D, F, G, V, X

Esempio di matrice (l'ordine è determinato dalla chiave primaria "CARGO"):

```
   A  D  F  G  V  X
A  C  A  R  G  O  b
D  d  e  f  h  i  j
F  k  l  m  n  p  q
G  s  t  u  v  w  x
V  y  z  0  1  2  3
X  4  5  6  7  8  9
```

La costruzione:
1. Prendere la chiave primaria "CARGO"
2. Scriverla eliminando i duplicati: CARGO
3. Completare con le lettere e le cifre rimanenti in ordine: CARGObdefhijklmnpqstuvwxyz0123456789

4. Disporre il tutto in una matrice 6×6

La sostituzione: Ogni carattere viene sostituito con una coppia di lettere ADFGVX

Esempi:
- a (riga A, colonna D) → AD
- t (riga G, colonna D) → GD
- c (riga A, colonna A) → AA
- k (riga F, colonna A) → FA

Il messaggio "attack" diventa → AD GD GD AD AA FA

Fase 2: La Trasposizione Colonnare

La chiave secondaria è una parola (es. "KEY")

1. Scrivere il testo sostituito sotto la chiave ripetuta:

```
K E Y
A D G
D G D
A D A
A F A
```

2. Ordinare le colonne alfabeticamente secondo la chiave (E < K < Y):

```
E K Y
D A G
G D D
D A A
F A A
```

3. Leggere per colonne:
Colonna E: D G D F
Colonna K: A D A A
Colonna Y: G D A A
```Il testo cifrato finale: DGDF ADAA GDAA
```

11.4 La Matematica del Sistema

Fase 1: La Sostituzione come Funzione

Matrice M: $(A \cup \{0\text{-}9\}) \rightarrow (ADFGVX)^2$

Ogni carattere p viene mappato alla coppia (r,c) dove M[r,c] = p

La cardinalità:
- Input: 36 caratteri (26 lettere + 10 cifre)
- Output: 36 coppie da 6 lettere = $6^2 = 36$ possibilità
- È una funzione biettiva ✓

La proprietà: È una sostituzione mono-alfabetica, ma con fractionation (ogni carattere diventa 2 simboli)

Fase 2: La Trasposizione come Permutazione

È identica al sistema colonnare classico (che vedremo nel Capitolo 12 sulle trasposizioni).

Per un messaggio di lunghezza L (dopo la sostituzione, sempre pari), con una chiave di lunghezza m:

Righe: $r = \lceil L/m \rceil$

La permutazione π è definita dall'ordine delle colonne.

Il sistema completo: È la composizione di due trasformazioni

$E = T \circ S$

dove S = la sostituzione (matrice), T = la trasposizione (colonnare)

L'inverso:

$D = S^{-1} \circ T^{-1}$
Per la decifratura: si inverte prima la trasposizione, poi si inverte la sostituzione.

11.5 Esempio Completo: "Attacco all'Alba"

Il messaggio: "ATTACKATDAWN" (12 caratteri)

La chiave primaria (matrice): "CARGO" (come sopra)

La chiave secondaria (trasposizione): "KEY"
Step 1: Rimozione degli spazi

"ATTACKATDAWN"

Step 2: La sostituzione con la matrice

```
a → AD, t → GD, t → GD, a → AD, c → AA, k → FA
a → AD, t → GD, d → DA, a → AD, w → GX, n → FG
```

Il testo sostituito (24 caratteri): ADGDGDADAAFAADGDDAADGXFG

Step 3: La scrittura sotto la chiave "KEY"

```
K E Y
A D G
D G D
A D A
A F A
A D G
D D A
A D G
X F G
```

Step 4: L'ordinamento delle colonne (E-K-Y)

```
E K Y
D A G
G D D
D A A
F A A
D A G
D D A
D A G
```

F X G
```

Step 5: La lettura per colonne
```
E: D G D F D D D F
K: A D A A A D A X
Y: G D A A G A G G
```

Il testo cifrato finale: DGDFDDDF ADADAADAX GDAAGAGG

(La spaziatura è per chiarezza; il testo veniva trasmesso senza spazi)

11.6 La Decifratura (con le chiavi note)

Input: DGDFDDDF ADADAADAX GDAAGAGG + le chiavi

Il processo inverso:

Step 1: L'inversione della trasposizione

Calcolare le dimensioni:
- Lunghezza totale: N = 24
- Lunghezza della chiave: m = 3
- Righe: r = ⌈24/3⌉ = 8

Dividere il testo cifrato in colonne (nell'ordine alfabetico E-K-Y):
- E (8 caratteri): DGDFDDDF
- K (8 caratteri): ADADAADAX
- Y (8 caratteri): GDAAGAGG

Riordinare secondo la chiave originale (K-E-Y):

```
K E Y
A D G
D G D
A D A
A F A
A D G
D D A
```
```

A D G
X F G

```
Leggere per righe: ADGDGDADAAFAADGDDAADGXFG

Step 2: L'inversione della sostituzione

Prendere il testo a coppie e cercare nella matrice:

AD → a, GD → t, GD → t, AD → a, AA → c, FA → k,
AD → a, GD → t, DA → d, AD → a, GX → w, FG → n

Il testo in chiaro: attackatdawn ✓
```

11.7 La Crittanalisi: La Sfida del 1918

Primavera 1918: L'offensiva tedesca Kaiserschlacht

La Germania lancia un massiccio attacco sul Fronte Occidentale (marzo-luglio 1918). Il cifrario ADFGVX entra in uso operativo.
Il Bureau du Chiffre (l'ufficio delle cifre francese) improvvisamente non riesce più a decifrare il traffico tedesco. Si verifica un blackout dell'intelligence totale.
Georges Painvin (1886-1980): Un capitano, ingegnere civile e crittoanalista brillante. Venne assegnato al problema dell'ADFGVX: doveva decifrare urgentemente il traffico per salvare la situazione strategica.

11.8 Il Metodo di Painvin: Anatomia di una Violazione

Fase 1: L'identificazione della struttura

L'osservazione iniziale: Tutto il traffico cifrato usa solo 6 lettere (A, D, F, G, V, X).

L'ipotesi: Si tratta di un sistema a due stadi (sostituzione + trasposizione). Massima profondità su pochi argomenti selezionati?

La conferma: I testi cifrati hanno sempre lunghezza pari → la sostituzione è 1:2 (ogni carattere diventa una coppia)

Fase 2: La determinazione della lunghezza della chiave di trasposizione

Il metodo: L'analisi delle coincidenze per determinare m.

La procedura:
1. Allineare il testo cifrato con se stesso a distanza d
2. Contare le coincidenze (le stesse lettere nelle stesse posizioni)
3. Tracciare un grafico delle coincidenze vs distanza

Il risultato atteso:
- Per le distanze d = multipli di m: coincidenze alte (le colonne sono ri-allineate)
- Per le distanze d ≠ multipli di m: coincidenze basse (distribuzione casuale)

Esempio quantitativo:

Testo di N = 240 caratteri

Test alla distanza d = 10:
- Coincidenze: 68/240 = 28,3%
- Atteso casuale (6 lettere): 1/6 ≈ 16,7%
- Differenza significativa! → d = 10 è probabilmente un multiplo di m

Test di altre distanze:
- d = 5: 65/240 = 27,1% (alto)
- d = 15: 62/240 = 25,8% (alto)
- d = 7: 42/240 = 17,5% (basso)
- d = 11: 39/240 = 16,3% (basso)

La conclusione: MCD(5, 10, 15) = 5 → m = 5

Fase 3: L'attacco sulla trasposizione (Anagramming)

Con m noto, il problema diventa: determinare la permutazione delle m colonne.

Il numero di permutazioni: m!

Per m = 5: 5! = 120 (fattibile!)

La procedura:

1. Dividere il testo in colonne assumendo m = 5
2. Provare tutte le 120 permutazioni
3. Per ogni permutazione:
   - Leggere per righe → testo "de-trasposto"
   - Verificare se ha la struttura corretta (solo coppie di 6 lettere)

- Calcolare le statistiche (le frequenze delle coppie)

4. La permutazione corretta produce statistiche "naturali"

L'identificazione della permutazione corretta:

Dopo la de-trasposizione, le coppie dovrebbero avere:
- Una distribuzione non uniforme (alcune coppie più comuni - corrispondono alle lettere tedesche comuni)
- Pattern riconoscibili

Fase 4: La ricostruzione della matrice di sostituzione

Con il testo de-trasposto (una sequenza di coppie ADFGVX):

1. L'analisi della frequenza delle coppie: Le coppie più frequenti corrispondono alle lettere comuni (E, N, I, R in tedesco)

2. Lo sfruttamento dei crib: I messaggi militari contengono formule standard:
   - "VONKOMMANDO" (dal comando)
   - "ANBEFEHLSHABER" (al comandante)
   - Coordinate note, nomi di luoghi

3. La ricostruzione parziale:
   - Identificare le coppie per le lettere note
   - Es: "VON" appare spesso → identificare le coppie corrispondenti a V, O, N
   - Cercare pattern nel testo cifrato

4. L'espansione iterativa: Ogni carattere identificato aiuta a identificare altri tramite il contesto

La complessità (metodo Painvin, manuale):
- Determinazione di m: giorni (l'analisi statistica era laboriosa)
- Test delle permutazioni: settimane (120 permutazioni × analisi manuale)
- Ricostruzione della matrice: settimane (con cribs limitati)

Totale: Painvin impiegò 3 mesi (marzo-giugno 1918)
Con un computer moderno: minuti

11.9 Il Successo e l'Impatto Storico

Giugno 1918: Painvin decifra l'ADFGVX

L'intelligence recuperata:
- I movimenti delle truppe tedesche
- I piani offensivi imminenti
- La logistica (munizioni, rifornimenti)

La conseguenza strategica:

La Seconda Battaglia della Marna (luglio 1918): I francesi, grazie all'intelligence Ultra precisa dall'ADFGVX decifrato, posizionarono le forze perfettamente per il contrattacco.
L'offensiva tedesca venne fermata. Iniziò il collasso tedesco che avrebbe portato all'armistizio (novembre 1918).

11.10 La Variante: ADFGX (Il Predecessore)

Prima dell'ADFGVX, la Germania usò ADFGX (5 lettere, matrice 5×5).

Le limitazioni:
- Solo l'alfabeto (25 lettere, I/J combinati)
- Nessuna cifra

Il motivo dell'upgrade: La necessità di includere le cifre per le coordinate, gli orari e le quantità militari.

L'aggiunta della lettera V → matrice 6×6 (36 caratteri) = ADFGVX

La sicurezza comparata:
- ADFGX: $5^2 = 25$ coppie possibili
- ADFGVX: $6^2 = 36$ coppie
- Aumento dell'entropia: $\log_2(36/25) \approx 0{,}53$ bit/carattere

11.11 Le Lezioni per la Crittografia Moderna

L'ADFGVX rappresenta diversi principi che rimarranno centrali:

1. La composizione di trasformazioni: Combinare la sostituzione con la trasposizione produce una sicurezza maggiore della somma delle parti (il principio dei cifrari prodotto moderni)

2. Il fractionation: Spezzare i caratteri in unità più piccole (qui: 1 carattere → 2 simboli) complica l'analisi statistica

3. La robustezza operativa: La scelta delle lettere per minimizzare gli errori di trasmissione - un engineering pratico, non solo teoria matematica

4. Il sistema a due chiavi: La chiave primaria (matrice) + la chiave secondaria (trasposizione) - un'anticipazione dei sistemi multi-chiave

5. Le vulnerabilità procedurali: Il sistema era teoricamente forte ma vulnerabile se:
- Le chiavi venivano usate troppo a lungo
- C'erano formule standard nei messaggi (cribs)
- Il volume del traffico era sufficiente per l'analisi statistica

La caduta dell'ADFGVX dimostrò che anche i sistemi sofisticati possono essere violati con:
- La determinazione
- Il genio matematico (Painvin)
- Risorse sufficienti (tempo, volume del traffico intercettato)

Ma anche che una crittografia robusta può resistere abbastanza a lungo da avere un impatto strategico (marzo-giugno 1918: 3 mesi di protezione prima della violazione).

CAPITOLO 12

I CIFRARI A TRASPOSIZIONE - TEORIA E CRITTANALISI

Mentre la maggior parte dell'attenzione storica si è concentrata sui cifrari di sostituzione (Cesare, Vigenère, ecc.), i cifrari a trasposizione - che riorganizzano le posizioni dei caratteri senza alterarne l'identità - hanno giocato un ruolo significativo, spesso in combinazione con la sostituzione (come nel caso dell'ADFGVX).

12.1 Il Principio Fondamentale

La trasposizione pura: Una permutazione π delle posizioni

La cifratura: $C[i] = P[\pi(i)]$

La proprietà matematica: π è una biezione su $\{0,1,...,N-1\}$

Le caratteristiche:
- Le frequenze sono preservate: $f_C(\alpha) = f_P(\alpha)$ per ogni α
- L'IC è preservato: $IC(C) = IC(P)$
- La lunghezza è preservata: $C = P$

L'implicazione crittanalitica: L'analisi di frequenza monocarattere è completamente inefficace. Gli attacchi devono sfruttare i pattern linguistici di ordine superiore (digrammi, trigrammi, parole).

12.2 La Trasposizione Colonnare Singola

Già incontrata nell'ADFGVX, formalizziamo ora il metodo completamente.

L'algoritmo:

1. Scegliere una chiave $K = k_1k_2...k\_m$ (una parola)
2. Ordinare le lettere della chiave alfabeticamente → permutazione σ
3. Scrivere il testo P in una matrice r × m (r righe, m colonne)
4. Leggere le colonne nell'ordine σ

Esempio:
Messaggio: "DEFENDTHEEASTWALLOFTHECASTLE" (28 caratteri)

Chiave: "ZEBRAS" (m = 6)

Ordine alfabetico: A B E R S Z → posizioni 5 6 2 3 4 1
Cioè: σ = [5,6,2,3,4,1] (la permutazione)

Matrice (r = ⌈28/6⌉ = 5):

Pos: 1 2 3 4 5 6
Key: Z E B R A S

D E F E N D
T H E E A S
T W A L L O
F T H E C A
S T L E _ _

Lettura per colonne nell'ordine A-B-E-R-S-Z (pos 5,2,6,3,4,1):

Col A (pos 5): N A L C _
Col B (pos 2): E H W T T
Col E (pos 6): D S O A _
Col R (pos 3): F E A H L
Col S (pos 4): E E L E _
Col Z (pos 1): D T T F S

Testo cifrato: NALC_ EHWTT DSO A_ FEAHL EELE_ DTTFS
```

(L'underscore rappresenta il padding, può essere X o può essere rimosso)

La formalizzazione: Per la posizione i nel testo in chiaro:

$\pi(i) = (i \bmod r) \times m + \sigma^{-1}(\lfloor i/r \rfloor)$

dove $r = \lceil N/m \rceil$, σ è la permutazione indotta dalla chiave.

## 12.3 La Trasposizione Colonnare Doppia

La motivazione: La singola trasposizione è vulnerabile all'anagramming.

La doppia trasposizione: Si applica il metodo due volte con chiavi diverse $K_1$, $K_2$

L'algoritmo:

1. Prima trasposizione con $K_1 \rightarrow C_1$
2. Seconda trasposizione su $C_1$ con $K_2 \rightarrow C_2$

Esempio:

```
Testo in chiaro: "ATTACKATDAWN"

Prima trasposizione (K_1 = "KEY", ordine 2-1-3):
K E Y
A T T → Col E: T C T W
A C K   Col K: A A A A
A T D   Col Y: T K D N
A W N

C_1 = "TCTW AAAA TKDN"

Seconda trasposizione (K_2 = "WORD", ordine 2-3-1-4):

W O R D
T C T W → Col O: C A K
A A A A   Col R: T A D
T K D N   Col W: T A T
```
```

Col D: W A N

C_2 = "CAK TAD TAT WAN"
```

Lo spazio delle chiavi: $K_1 \times K_2$

Per chiavi lunghe 8 caratteri: $8! \times 8! = (40320)^2 \approx 1{,}6 \times 10^9$

La sicurezza: Il singolo anagramming è inefficace (è necessario conoscere entrambe le chiavi).

L'attacco richiede tecniche avanzate o un testo molto lungo con dei cribs.

## 12.4 Il Railfence Cipher

Il principio: Scrivere il messaggio a zig-zag su "binari", leggere per binari.

Esempio (3 binari):

```
Messaggio: "WEAREDISCOVEREDFLEEATONCE"

Binario 1: W...E...C...R...L...T...E
Binario 2: .E.R.D.S.O.E.E.F.E.A.O.C.
Binario 3: ..A...I...V...D...E...N..

Lettura:
Binario 1: WECRLTE
Binario 2: ERDSOEEFEAOC
Binario 3: AIVDEN
Testo cifrato: WECRLTE ERDSOEEFEAOC AIVDEN
```

La generalizzazione: n binari

La periodicità: 2(n-1) caratteri per un ciclo zig-zag completo

La sicurezza:

Lo spazio delle chiavi: solo n (il numero di binari) → tipicamente $n \in \{2,3,...,10\}$
```

K ≈ 9 (ridicolmente piccolo!)

È vulnerabile alla forza bruta immediata.

L'uso: Principalmente didattico o combinato con altri metodi.

12.5 La Crittanalisi: L'Anagramming

Il principio: Sfruttare i vincoli linguistici per ricostruire l'ordine originale.
L'approccio Hill-Climbing:

```
Input: testo_cifrato C
Output: permutazione π

1. π ← permutazione casuale iniziale
2. Ripeti:
3.    T ← applica π⁻¹ a C (tentativo di ricostruzione)
4.    score ← valuta_leggibilità(T)
5.    π' ← perturba π (swap casuale di 2 posizioni)
6.    T' ← applica π'⁻¹ a C
7.    score' ← valuta_leggibilità(T')
8.    Se score' > score:
9.       π ← π'  // accetta il miglioramento
10. Fino a convergenza
11. Return π
```

La funzione di scoring:
score(T) = Σ_{digramma d in T} w(d)

dove w(d) = peso basato sulla frequenza nella lingua

Esempi di pesi:
TH: w = 3,56
HE: w = 3,07
IN: w = 2,43
...
ZQ: w = 0,001 (molto raro)
```

La complessità:
```

Per k colonne:
- Spazio di ricerca: k!
- Hill-climbing: O(k² × iterazioni)

Con k=10, iter=1000: O(100.000) valutazioni

È fattibile computazionalmente in secondi (con metodi moderni).
I vincoli linguistici aggiuntivi:

- Il dizionario: Verificare se la ricostruzione contiene parole valide
- Gli N-grammi: I trigrammi comuni (THE, AND, ING in inglese)
- La grammatica: I pattern POS (part-of-speech) plausibili

12.6 La Trasposizione Irregolare

La motivazione: La trasposizione regolare (colonnare) ha un pattern geometrico riconoscibile.

La trasposizione irregolare: Una permutazione arbitraria senza struttura.

La definizione:

La chiave: una permutazione esplicita π = [π(0), π(1), ..., π(N-1)]

La cifratura: C[i] = P[π(i)]

Esempio:
```
Messaggio (N=12): "ATTACKATDAWN"

Permutazione: [11, 3, 7, 0, 9, 2, 5, 1, 10, 4, 8, 6]

C[0] = P[11] = N
C[1] = P[3] = A
C[2] = P[7] = T
...

Testo cifrato: "NATAATKTWCDA"
```

Lo spazio delle chiavi: N!

Per N=20: $20! \approx 2{,}4 \times 10^{18}$

La gestione della chiave:

Il problema: Comunicare una permutazione di N elementi è laborioso.

Soluzione 1: Derivare π da una parola chiave
Esempio: La chiave "SECRET"
Ordinare le lettere: CEERTS → posizioni relative
Usare come inizio della permutazione, completare con le posizioni rimanenti

Soluzione 2: Una funzione pseudo-random con seed

π(i) = f(i, seed) dove f è una funzione deterministica nota

La debolezza:

Anche la trasposizione irregolare preserva l'IC e le frequenze → è vulnerabile agli attacchi statistici sofisticati con testi lunghi e cribs.

PARTE II: L'ERA DELLE MACCHINE CIFRANTI (1918-1945)

CAPITOLO 13

ENIGMA - L'APICE DELLA CRITTOGRAFIA ELETTROMECCANICA

Nel caos che seguì la Prima Guerra Mondiale, mentre l'Europa contava le proprie ferite e cercava di ricostruire una parvenza di ordine, un ingegnere tedesco stava per cambiare per sempre il volto della crittografia. Arthur Scherbius, un brillante ingegnere elettrico formatosi alla Technische Hochschule di Monaco, aveva una visione: una macchina che potesse cifrare i messaggi con una velocità e una sicurezza senza precedenti, liberando gli operatori dalla fatica e dagli errori della cifratura manuale. La sua creazione, presentata nel 1923 e perfezionata negli anni successivi, avrebbe dominato le comunicazioni segrete per due decenni e avrebbe influenzato il corso della Seconda Guerra Mondiale in modi che solo decenni dopo sarebbero stati pienamente compresi.

13.1 Arthur Scherbius e la Rivoluzione Meccanica

Arthur Scherbius (1878-1929) incarnava lo spirito dell'ingegneria tedesca del primo Novecento: una formazione rigorosa, un approccio sistematico ai problemi e un'ossessione per la precisione e l'affidabilità. Dopo la laurea nel 1903, fondò nel 1918

(proprio mentre la Germania collassava militarmente) la Scherbius & Ritter a Berlino, un'azienda dedicata allo sviluppo di dispositivi elettromeccanici innovativi.
La guerra aveva dimostrato brutalmente l'inadeguatezza dei sistemi crittografici tradizionali. Il cifrario di Vigenère era caduto sotto i colpi di Kasiski e Friedman. L'ADFGVX, pur essendo sofisticato, richiedeva tempo prezioso ed era vulnerabile agli errori umani sotto stress. I nomenclatori erano lenti e richiedevano codebook voluminosi, vulnerabili alla cattura. Serviva qualcosa di radicalmente nuovo.
Scherbius concepì una macchina basata su rotori - dei dischi contenenti cablaggi elettrici che implementavano permutazioni complesse. L'idea non era completamente nuova (il principio dei rotori era stato esplorato da altri inventori, tra cui l'olandese Hugo Koch e lo svedese Arvid Damm), ma Scherbius la perfezionò in un dispositivo pratico e commercialmente valido.
Il primo brevetto fu depositato il 23 febbraio 1918 in Germania (numero 416,219), seguito da brevetti nel Regno Unito (1919) e negli Stati Uniti (1928). Il dispositivo venne presentato pubblicamente nel 1923 al Congresso dell'Unione Postale Internazionale.

13.2 L'Evoluzione di Enigma: Dalle Origini Commerciali all'Adozione Militare

Enigma A (1923-1924): La prima versione commerciale, pesante circa 50 kg, con una configurazione a 4 rotori, destinata alle banche e alle aziende. Era costosa, ingombrante e le vendite furono modeste.

Enigma B (1924): Un miglioramento cruciale - l'introduzione del riflettore (Umkehrwalze), il componente che avrebbe definito il carattere distintivo di Enigma. Il riflettore rendeva il processo involutorio: la cifratura e la decifratura usavano la stessa configurazione, un'enorme semplificazione operativa.

Enigma C (1926): Una versione compatta, con il peso ridotto a ~12 kg, adottata inizialmente dalla Reichsmarine (la Marina tedesca). Qui inizia la transizione dal mercato civile a quello militare.

Enigma I (Wehrmacht, 1930): La versione che avrebbe fatto la storia. Tre rotori selezionabili da un set di cinque, il riflettore, e l'innovazione cruciale: il Plugboard (Steckerbrett) - un pannello con 26 jack che permetteva connessioni a coppie, aggiungendo uno strato ulteriore di complessità.

La Reichswehr (l'esercito tedesco) iniziò le valutazioni serie nel 1926. Le considerazioni erano pratiche e strategiche:

- Velocità: Cifratura e decifratura in tempo reale semplicemente digitando
- Sicurezza: Una complessità matematica senza precedenti

- Standardizzazione: La stessa macchina per tutte le unità, semplificando la logistica
- Robustezza: La meccanica tedesca era affidabile, funzionava anche in condizioni difficili

Nel 1930, Enigma divenne lo standard per la Wehrmacht. Con l'ascesa del nazismo e la massiccia espansione militare (1933-1939), la produzione esplose: circa 100.000 unità prodotte entro il 1945, distribuite a ogni livello - dai comandi supremi alle singole unità tattiche, dalla Luftwaffe agli U-Boot, dalle SS ai servizi di intelligence.

13.3 L'Anatomia di Enigma: La Macchina nel Dettaglio

Immaginare Enigma come una semplice macchina da scrivere crittografica sarebbe riduttivo. Era un capolavoro di ingegneria elettromeccanica, dove ogni componente contribuiva a creare una trasformazione crittografica di complessità straordinaria.

La Tastiera (Tastatur): 26 tasti disposti nel layout QWERTZ tedesco (simile al QWERTY ma con la Y e la Z scambiate). Ogni pressione chiudeva un circuito elettrico che attraversava l'intera macchina prima di illuminare una lampadina sul pannello visualizzatore.

Il Plugboard (Steckerbrett): Una delle innovazioni militari più significative. 26 jack disposti in una configurazione 2×13 sulla parte frontale della macchina. Dieci cavi di connessione (Stecker) permettevano di collegare 20 lettere a coppie, lasciando 6 lettere "unchanged" (non connesse). Ogni coppia implementava una trasposizione semplice: se A era connesso a R, allora A$\leftrightarrow$R.

La funzione: una permutazione aggiuntiva applicata prima e dopo i rotori, aumentando drammaticamente lo spazio delle chiavi.

I Rotori (Walzen): Il cuore crittografico di Enigma. Erano dischi di bachelite (un materiale isolante) del diametro di circa 10 cm e dello spessore di 1,5 cm. Su ciascuna faccia c'erano 26 contatti elettrici (corrispondenti alle 26 lettere). All'interno, il cablaggio implementava una permutazione fissa - ogni contatto di ingresso era connesso a un contatto di uscita diverso.

La Wehrmacht standard disponeva di 5 rotori (numerati I, II, III, IV, V), ciascuno con un cablaggio unico. Per ogni operazione, si sceglievano 3 rotori e si disponevano in un ordine specifico (es. II-V-III).

Ogni rotore aveva:

- L'Anello esterno (Ring): Regolabile, con 26 posizioni numerate 01-26 o letterate A-Z. Ruotava rispetto al cablaggio interno, aggiungendo un offset alla trasformazione.

- La Tacca (Notch): Un piccolo dente meccanico in una posizione specifica che causava lo "stepping" del rotore successivo quando veniva raggiunto. La posizione della tacca variava per ogni rotore:
  - Rotore I: tacca in Q
  - Rotore II: tacca in E
  - Rotore III: tacca in V
  - Rotori IV, V: tacche in J e Z rispettivamente

Il Riflettore (Umkehrwalze - UKW): Un disco simile ai rotori ma fisso (non ruotava). Era cablato con 13 coppie di trasposizioni, implementando una permutazione involutoria: se A→Y, allora Y→A. Le versioni principali erano: UKW-A, UKW-B, UKW-C, ciascuna con un cablaggio diverso.

La funzione del riflettore era geniale nella sua semplicità: dopo aver attraversato i tre rotori in una direzione, il segnale veniva "riflesso" indietro attraverso gli stessi rotori (ma in ordine inverso e con trasformazioni inverse), creando una simmetria perfetta.

La conseguenza fondamentale: cifratura ≡ decifratura. La stessa configurazione serviva per entrambe le operazioni - un'enorme semplificazione operativa.

Il Rotore di Ingresso (Eintrittswalze - ETW): Un componente spesso trascurato ma necessario. Era la connessione diretta tra il plugboard e i rotori: contatto i → contatto i (identità). Era presente in tutte le versioni.

Il Lampboard (Lampenfeld): Un pannello con 26 lampadine disposte come la tastiera. Quando un tasto veniva premuto, una lampadina si illuminava indicando la lettera cifrata (o decifrata). L'operatore trascriveva la lettera illuminata.

13.4 Il Percorso del Segnale: Seguendo l'Elettricità

Comprendere Enigma richiede di seguire fisicamente il percorso del segnale elettrico dal tasto premuto alla lampada illuminata. Consideriamo la cifratura della lettera A:

1. La pressione del tasto A: Si chiude il circuito elettrico. La corrente fluisce dalla batteria.

2. Il Plugboard (andata): Se A è connesso a R tramite un cavo, il segnale viene deviato alla posizione R (17). Altrimenti, rimane nella posizione A (0).

3. L'ETW: Connessione diretta. La posizione 17 → il contatto 17 sul primo rotore.

4. Il Rotore Destro (es. Rotore III in posizione B): Il rotore ha ruotato di 1 posizione rispetto alla posizione neutrale. La trasformazione tiene conto dell'offset. Se il cablaggio interno mappa il contatto 18 (17+1 per l'offset) al contatto 22, il segnale esce al contatto 22-1=21.

5. Il Rotore Medio (es. Rotore II in posizione A): Riceve il segnale al contatto 21, applica la sua permutazione interna, emette al contatto (esempio) 5.

6. Il Rotore Sinistro (es. Rotore I in posizione A): Riceve il segnale al contatto 5, applica la permutazione, emette al contatto (esempio) 6.

7. Il Riflettore (UKW-B): Riceve al contatto 6 (G), riflette al contatto 11 (L) secondo il cablaggio fisso.

8. Il ritorno attraverso i rotori: Il segnale ora attraversa i rotori in ordine inverso:
- Rotore Sinistro (inverso): Contatto 11 → ...
- Rotore Medio (inverso): ...
- Rotore Destro (inverso): ... → contatto (esempio) 16

9. L'ETW: Connessione diretta di ritorno.

10. Il Plugboard (ritorno): Se la posizione 16 (Q) non è connessa, rimane Q. Altrimenti, attraversa la trasposizione.

11. La lampada Q si illumina: L'operatore vede Q, lo trascrive come lettera cifrata.

Il risultato: A → Q

Questo percorso complesso, attraversando 7 trasformazioni (plugboard + 3 rotori andata + riflettore + 3 rotori ritorno + plugboard), crea una permutazione di complessità straordinaria.

13.5 Il Meccanismo di Stepping: L'Orologio Crittografico

Enigma non sarebbe stato rivoluzionario se i rotori fossero rimasti statici. Il genio del sistema risiede nel meccanismo di stepping automatico che faceva avanzare i rotori dopo ogni lettera cifrata, cambiando continuamente la trasformazione.

Il principio base (l'analogia con l'odometro meccanico):

- Il rotore destro (veloce): Avanza di 1 posizione dopo ogni lettera cifrata
- Il rotore medio: Avanza quando il rotore destro raggiunge la sua tacca (notch)

- Il rotore sinistro (lento): Avanza quando il rotore medio raggiunge la sua tacca

Esempio di sequenza di stepping:

Configurazione iniziale: rotori I-II-III, posizioni A-A-A

- Lettera 1: A-A-A → step → A-A-B (solo il destro)
- Lettera 2: A-A-B → step → A-A-C
- ...
- Lettera 21: A-A-U → step → A-A-V (III ha la tacca in V)
- Lettera 22: A-A-V → step → A-B-W (il destro avanza + il medio avanza per la tacca)
- Lettera 23: A-B-W → step → A-B-X
- ...

L'anomalia del Double-stepping: Un fenomeno critico spesso ignorato. Quando il rotore medio è sulla propria tacca:

- Causa lo stepping del rotore sinistro (comportamento normale)
- Ma avanza anche se stesso (double step)

Esempio:

Configurazione: I-II-III, posizioni A-D-V

- Lettera N: A-D-V → step → A-E-W (il medio avanza per la tacca del destro)
- Lettera N+1: A-E-W → step → A-E-X
- ...

Quando il medio raggiunge E (la tacca del II):

A-E-? → step → B-F-? (il sinistro avanza + il medio avanza se stesso!)

Questa irregolarità era una debolezza del sistema - introduceva una periodicità anomala sfruttabile dai crittoanalisti. Ma la Wehrmacht non ne era consapevole.

La periodicità totale: Con 3 rotori, teoricamente ci sono $26^3 = 17.576$ posizioni. Con il double-stepping e le tacche, il periodo effettivo è < 16.900 (irregolare). Per i messaggi tipici (< 250 caratteri), il rotore sinistro raramente avanzava - effettivamente, il sistema operava come un polialfabetico con un periodo variabile tra 26 e 650.

13.6 La Matematica dei Rotori: Permutazioni e Gruppi

Possiamo formalizzare Enigma come una composizione di permutazioni sul gruppo simmetrico S_{26} (le permutazioni di 26 elementi).

Il rotore come permutazione: Ogni rotore R implementa σ_R: $\mathbb{Z}_{26} \to \mathbb{Z}_{26}$

Esempio - Rotore I (Wehrmacht):

Input: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Output: 4 10 12 5 11 6 3 16 21 25 13 19 14 22 24 7 23 20 18 15 0 8 1 17 2 9
E K M F L G D Q V Z N T O W Y H X U S P A I B R C J
```

σ_I(0) = 4 (A→E), σ_I(1) = 10 (B→K), ecc.

La posizione del rotore: L'offset n ∈ {0,1,...,25}

La permutazione effettiva con l'offset n:

σ_{R,n}(x) = (σ_R((x + n) mod 26) - n) mod 26

La spiegazione:
1. Compensare la rotazione: x + n
2. Applicare la permutazione interna: σ_R(x + n)
3. Compensare inversamente: risultato - n

Il Ring setting: Un offset aggiuntivo r ∈ {1,...,26}

L'effetto: sposta la tacca di r-1 posizioni, modifica leggermente la trasformazione.

La formula completa:

σ_{R,n,r}(x) = (σ_R((x + n - r + 1) mod 26) - n + r - 1) mod 26

Il Riflettore: Una permutazione involutoria ρ: $\mathbb{Z}_{26} \to \mathbb{Z}_{26}$

La proprietà: ρ(ρ(x)) = x per ogni x

Matematicamente: ρ è la composizione di 13 trasposizioni disgiunte (coppie).

Esempio UKW-B:
```

```
A↔Y, B↔R, C↔U, D↔H, E↔Q, F↔S, G↔L, I↔P, J↔X, K↔N, M↔O, T↔V, W↔Z
```

Il Plugboard: Un'altra permutazione involutoria π_S

Con 10 coppie: 20 lettere trasposte, 6 unchanged.

Il sistema completo: Per le posizioni dei rotori (n_L, n_M, n_R):

$$E(x) = \pi_S \circ \sigma_L^{-1} \circ \sigma_M^{-1} \circ \sigma_R^{-1} \circ \rho \circ \sigma_R \circ \sigma_M \circ \sigma_L \circ \pi_S(x)$$

Dove:
- π_S è applicato due volte (andata e ritorno)
- σ_R, σ_M, σ_L in andata
- ρ riflette
- σ_R^{-1}, σ_M^{-1}, σ_L^{-1} in ritorno (le permutazioni inverse)

La proprietà involutoria globale: $E(E(x)) = x$

La dimostrazione (schema): Ogni componente è involutorio (plugboard, riflettore) o ha il suo inverso nel percorso di ritorno (rotori). La composizione risulta involutoria.

La conseguenza pratica: La stessa configurazione per la cifratura e la decifratura ✓

La conseguenza crittografica critica: $E(x) \neq x$ per ogni x

Nessuna lettera può essere cifrata in se stessa! Questo deriva dalla proprietà involutoria del riflettore. Se esistesse un x tale che $E(x) = x$, significherebbe che il segnale ritorna alla stessa posizione - impossibile perché il riflettore mappa sempre $i \leftrightarrow j$ con $i \neq j$.

Questa proprietà diventerà il tallone d'Achille di Enigma.

13.7 Lo Spazio delle Chiavi: Un Universo Combinatorio

La sicurezza di Enigma non risiedeva in un singolo numero enorme, ma nella combinazione moltiplicativa di molteplici parametri indipendenti. Analizziamoli sistematicamente:

1. La selezione e l'ordine dei rotori (Wehrmacht 3-rotori, set di 5)

Le permutazioni di 3 da 5: P(5,3) = 5!/(5-3)! = 5×4×3 = 60

2. I Ring settings (Ringstellung)

Ogni rotore: 26 posizioni possibili
Tre rotori: $26^3 = 17.576$

3. Le posizioni iniziali (Grundstellung)

Visibili attraverso le finestrelle sulla macchina.
Tre rotori: $26^3 = 17.576$

4. Il Plugboard

10 coppie da 26 lettere.

Il calcolo:
```
Prima coppia: C(26,2) = 325 scelte
Seconda coppia: C(24,2) = 276 scelte
...
Decima coppia: C(8,2) = 28 scelte
Totale ordinato: 325×276×231×190×153×120×91×66×45×28

Ma l'ordine delle coppie è irrilevante: dividere per 10!

Numero di configurazioni del plugboard = 150.738.274.937.250

(Circa 1,5 × 10^14)
```

Lo spazio totale Wehrmacht (3 rotori):

$K_{total} = 60 \times 17.576 \times 17.576 \times 150.738.274.937.250$

$K_{total} \approx 1,07 \times 10^{23}$

Per confronto:
- Vigenère (chiave 10 lettere): $26^{10} \approx 1,4 \times 10^{14}$
- Enigma: 10^{23} - circa 1 miliardo di volte più grande!

La variante Kriegsmarine M3 (3 rotori da un set di 8):

P(8,3) = 8×7×6 = 336 (invece di 60)
K_total ≈ 6×10^{23}

La variante U-Boot M4 (4 rotori, introdotta nel febbraio 1942):

4 rotori con un riflettore "sottile"

Moltiplicazione ulteriore per un fattore ~26 (il rotore aggiuntivo)

K_total ≈ $2{,}8 \times 10^{25}$

Questi numeri rappresentavano, negli anni '30-'40, una sicurezza praticamente assoluta contro la forza bruta. Anche testando 1 milione di configurazioni al secondo, lo spazio di Enigma richiederebbe 3×10^{9} anni per essere esaurito - ordini di grandezza superiori all'età dell'universo!
Ma, come la storia avrebbe dimostrato, uno spazio delle chiavi enorme non garantisce la sicurezza se esistono debolezze strutturali sfruttabili.

13.8 La Procedura Operativa Wehrmacht: Teoria e Pratica

La sicurezza di un sistema crittografico non dipende solo dalla robustezza matematica dell'algoritmo, ma criticamente dalle procedure operative - come le chiavi vengono distribuite, gestite e utilizzate sul campo. Enigma, per quanto sofisticato tecnicamente, fu infine compromesso tanto per le debolezze procedurali quanto per le vulnerabilità matematiche.

13.8.1 Il Sistema di Distribuzione delle Chiavi

La Wehrmacht distribuiva mensilmente ai comandi lo Schlüsselmaterial (il materiale delle chiavi) - documenti stampati su carta solubile in acqua (per la distruzione rapida se catturati) contenenti le configurazioni giornaliere.

Esempio di foglio delle chiavi - Maggio 1940:

GEHEIM! (SEGRETO!)
Schlüssel für Heer und Luftwaffe - Mai 1940

Tag	Walzenlage	Ringstellung	Steckerverbindungen	Kenngruppen
1	II-IV-V	24-13-22	AV BS CG DL FU HZ IN KM OW RX	WXC PLM TGH
2	I-III-IV	06-18-07	AY BR CU DH EQ FW GI KP LZ OX	MNK QRS VWZ
3	III-I-II	03-26-14	AN BD CO EI FK GP HT LW MU XY	ABC DEF GHI

...

15 II-V-III 06-18-07 AP BR CM DZ EJ FQ GY HX IV KW WXC TGH PLM
...

I componenti della configurazione giornaliera (Grundschlüssel):

- Walzenlage (disposizione dei rotori): Quali 3 rotori e in che ordine. Es: II-V-III significa Rotore II (sinistro), Rotore V (medio), Rotore III (destro)

- Ringstellung (posizione degli anelli): Il setting per ciascun rotore (01-26 o A-Z). Es: 06-18-07 = F-R-G. Questo parametro veniva cambiato mensilmente o settimanalmente, raramente giornalmente (pigrizia operativa!)

- Steckerverbindungen (connessioni del plugboard): 10 coppie di lettere. Es: AP-BR-CM significa A↔P, B↔R, C↔M, ecc.

- Kenngruppen (gruppi discriminatori): Trigrammi casuali usati per identificare quale chiave giornaliera era stata usata, facilitando il riconoscimento da parte del destinatario.

Tutti gli operatori dello stesso network (es. tutte le unità Heer del settore Ovest) ricevevano lo stesso Schlüsselmaterial. Il giorno 15 di maggio, tutti avrebbero configurato le loro macchine identicamente secondo il foglio.

13.8.2 La Procedura di Cifratura (Sistema Pre-1940)

La configurazione giornaliera era solo il primo livello. Per ogni singolo messaggio, l'operatore doveva scegliere una chiave messaggio (Spruchschlüssel) - le posizioni iniziali dei rotori per quel messaggio specifico.

La procedura completa:

Passo 1: L'operatore mittente configura Enigma secondo il Grundschlüssel del giorno (es. 15 maggio 1940: II-V-III, ring F-R-G, plugboard AP-BR-CM-...)

Passo 2: L'operatore sceglie una posizione iniziale casuale (Grundstellung messaggio), es. QWE. Imposta i rotori su Q-W-E (visibile nelle finestrelle).

Passo 3: L'operatore sceglie una chiave messaggio casuale, es. RTZ (tre lettere scelte arbitrariamente - qui iniziano i problemi di sicurezza operativa).

Passo 4: Con i rotori su QWE, l'operatore cifra RTZ due volte: RTZRTZ

Supponiamo che l'output sia: KIVBJE

Perché doppio? Teoria: protezione contro gli errori di trasmissione. Se una singola lettera veniva corrotta, il confronto tra la prima e la seconda tripla avrebbe permesso la correzione. Pratica: questa ridondanza diventerà una vulnerabilità catastrofica.

Passo 5: L'operatore trasmette in chiaro:
- Il Grundstellung usato: QWE (in chiaro o secondo convenzione)
- L'indicatore cifrato: KIVBJE

Poi imposta i rotori su RTZ (la chiave messaggio vera) e cifra il messaggio effettivo.

Passo 6: Il messaggio completo trasmesso:

```
QWE KIVBJE LXFOPVEFRNHR...
└┬┘ └──┬──┘ └─────┬──────
 │     │          └─ messaggio cifrato con la chiave RTZ
 │     └──────────── chiave messaggio cifrata con QWE
 └────────────────── grundstellung (talvolta omesso, per convenzione)
```

13.8.3 La Ricezione e la Decifratura

Passo 1: Il destinatario riceve il messaggio, vede l'indicatore KIVBJE

Passo 2: Configura la sua Enigma con il Grundschlüssel del giorno (identico al mittente)

Passo 3: Imposta i rotori su QWE (il grundstellung comunicato)

Passo 4: Digita KIVBJE → la macchina produce RTZRTZ

Passo 5: Estrae la chiave messaggio: RTZ (confronta la prima e la seconda tripla per verificare - se identiche, nessun errore di trasmissione)

Passo 6: Imposta i rotori su RTZ

Passo 7: Digita il resto del messaggio cifrato → ottiene il testo in chiaro

Il sistema era elegante nella teoria, ma conteneva tre vulnerabilità critiche che i polacchi avrebbero sfruttato magistralmente.

13.9 Le Tre Debolezze Fatali

Debolezza 1: L'Indicatore Doppio

Cifrare RTZRTZ con lo stesso grundstellung QWE produceva correlazioni matematiche precise:

- La lettera in posizione 1 (R) e la lettera in posizione 4 (R) cifrano la stessa lettera con configurazioni dei rotori distanti esattamente 3 step
- La stessa relazione vale per le posizioni 2↔5 e 3↔6

Se il testo cifrato è KIVBJE:
- K e B cifrano entrambi R (con 3 step di differenza)
- I e J cifrano entrambi T
- V e E cifrano entrambi Z

Questa regolarità, osservata su centinaia di messaggi giornalieri, permetteva la costruzione di "catene" che rivelavano la struttura delle permutazioni - esattamente il metodo che Marian Rejewski avrebbe sviluppato.

Debolezza 2: La Procedura Standardizzata

Tutti gli operatori seguivano la stessa procedura. La Wehrmacht non prevedeva variazioni - una rigidità che facilitava l'addestramento ma creava pattern riconoscibili.

Debolezza 3: Le Scelte Non Casuali

Gli operatori umani sono terribilmente non casuali. Quando veniva chiesto di "scegliere tre lettere a caso", gli operatori tendevano a:
- Usare AAA, BBB, ABC, XYZ (pattern evidenti)
- Usare le iniziali proprie o della fidanzata
- Usare sequenze sulla tastiera (QWE, ASD)
- Ripetere le stesse chiavi giornalmente (pigrizia)

Questa non-casualità umana forniva punti di ingresso per gli attacchi.

13.8.4 La Riforma del Maggio 1940

Dopo che i polacchi comunicarono agli alleati (luglio 1939) di aver violato il sistema, e dopo le prime settimane di guerra dove i tedeschi sospettarono una compromissione, la Wehrmacht modificò la procedura (15 maggio 1940):
Il nuovo sistema:
- Un Grundstellung giornaliero standard per tutti (es. sempre WXC)

- L'operatore sceglie la chiave messaggio (es. RTZ)
- Cifra RTZ una sola volta (non più doppio)
- Trasmette RTZ cifrato + il messaggio

I vantaggi:
- Elimina la correlazione dell'indicatore doppio
- Mantiene la semplicità operativa

Il nuovo problema:
- Il Grundstellung standard è visibile/prevedibile
- Se il Grundstellung è sempre uguale, la stessa chiave messaggio produce lo stesso indicatore - i pattern sono ancora sfruttabili

La riforma fu un miglioramento, ma non sufficiente. I metodi di Bletchley Park (i test dei cribs, le bombe di Turing) erano già indipendenti dall'indicatore doppio, basandosi su debolezze strutturali più profonde.

13.10 La Crittanalisi Polacca: Davide contro Golia (1932-1939)

La Polonia, risorta come nazione indipendente dopo secoli di spartizioni, si trovava negli anni '20-'30 in una posizione precaria: schiacciata tra una Germania revanchista a ovest e un'Unione Sovietica ostile a est. La sopravvivenza nazionale dipendeva da un'intelligence superiore. Se non potevi vincere con la forza militare bruta, dovevi vincere con l'intelligenza - letteralmente.

13.10.1 Il Biuro Szyfrów e la Decisione Strategica

Il Biuro Szyfrów (l'Ufficio delle Cifre), fondato nel 1919 come parte dello Stato Maggiore polacco, aveva una missione duplice: proteggere le comunicazioni polacche e violare quelle nemiche. Negli anni '20, i metodi tradizionali (linguisti, esperti classici) permettevano la decifratura dei cifrari diplomatici convenzionali.
Ma nel 1926, quando la Germania iniziò ad adottare Enigma, tutto cambiò. Gli esperti linguistici del Biuro si trovarono impotenti davanti a un traffico cifrato che non mostrava pattern linguistici riconoscibili. L'analisi di frequenza, l'anagramming, i metodi di Kasiski - tutto era inutile.
Il maggiore Gwido Langer, capo del Biuro, prese una decisione audace: il problema era matematico, servivano dei matematici.

13.10.2 Il Reclutamento: Poznań 1929

L'Università di Poznań, una città a 300 km da Berlino, aveva un eccellente dipartimento di matematica. Il professor Zdzisław Krygowski, su richiesta del Biuro, organizzò nel 1929 un corso segreto di crittografia per studenti selezionati.

Tra oltre 20 studenti, tre si distinsero:

Marian Rejewski (1905-1980): Nato a Bromberg (oggi Bydgoszcz), laureato in matematica nel 1929. Era brillante nella teoria dei numeri e nell'algebra astratta - competenze che si sarebbero rivelate cruciali. Tranquillo, metodico, incredibilmente tenace.
Jerzy Różycki (1909-1942): Specializzato in matematica applicata. Era eccellente nell'implementazione pratica delle teorie astratte. Sarebbe morto tragicamente in un naufragio nel 1942.

Henryk Zygalski (1908-1978): Esperto in geometria e combinatoria. Avrebbe sviluppato il metodo dei "fogli perforati" che avrebbe portato il suo nome.

Nell'estate del 1932, tutti e tre furono assunti a tempo pieno dal Biuro Szyfrów, Sezione BS-4 (Germania). Il loro compito: violare Enigma.

13.10.3 Il Materiale di Partenza: L'Alleato Francese

I polacchi non partivano da zero totale. Nel 1931, un colpo di fortuna cambiò tutto.
Hans-Thilo Schmidt, un tedesco impiegato al Chiffrierstelle (l'ufficio delle cifre della Wehrmacht) di Berlino, scontento e in difficoltà finanziarie, offrì ai francesi dei documenti segreti in cambio di denaro. Il suo nome in codice era "Asché" (la francesizzazione delle sue iniziali H.E.).

Schmidt fornì:
- I manuali operativi di Enigma
- Le chiavi giornaliere (Schlüsselmaterial) per alcuni mesi
- Le procedure operative dettagliate
- Il cablaggio dei rotori I, II, III
- Le specifiche del riflettore UKW-A

Il Service de Renseignement français, sotto il capitano Gustave Bertrand, esaminò il materiale ma lo giudicò di scarso valore - non capirono come usarlo. Bertrand, più perspicace, pensò agli alleati polacchi.
Nel dicembre 1931, Bertrand viaggiò a Varsavia e consegnò il materiale al maggiore Langer. Per i polacchi fu una rivelazione: finalmente conoscevano il nemico dall'interno.
Tuttavia conoscere il cablaggio dei rotori non bastava. Ogni giorno, la configurazione cambiava (l'ordine dei rotori, i ring settings, il plugboard). Il problema rimaneva: come determinare la configurazione giornaliera dal traffico intercettato?

13.10.4 L'Intuizione di Rejewski: La Teoria delle Permutazioni

Rejewski, assegnato al problema full-time dal settembre 1932, ebbe un'intuizione brillante che avrebbe cambiato la storia: non attaccare Enigma come un linguista, ma come un matematico.

L'idea centrale: Sfruttare la vulnerabilità dell'indicatore doppio non per decifrare direttamente, ma per rivelare la struttura matematica delle permutazioni implementate da Enigma.
Ogni giorno, centinaia di messaggi intercettati iniziavano con indicatori cifrati a 6 lettere (es. KIVBJE). Rejewski osservò:

Se l'indicatore è KIVBJE, rappresenta la cifratura di RTZRTZ (o qualche altra coppia ripetuta).

Le posizioni 1 e 4: entrambe cifrano la stessa lettera (R) ma con i rotori in configurazioni distanti 3 step

- Pos 1: K è la cifratura di R con i rotori nella posizione iniziale grundstellung
- Pos 4: B è la cifratura di R con i rotori avanzati di 3 posizioni

Questo definisce una relazione tra K e B: esiste una trasformazione che mappa K→B, determinata dalla configurazione Enigma del giorno.

Matematicamente: Definire una permutazione S tale che S(K) = B

Raccogliendo dati da molti messaggi dello stesso giorno:
- Messaggio 1: indicatore KIVBJE → K→B
- Messaggio 2: indicatore HNMPLQ → H→P
- Messaggio 3: indicatore FXGTRA → F→T
- ...

Dopo aver raccolto dozzine di coppie, Rejewski poteva costruire la permutazione S completamente o quasi.

La proprietà cruciale: S dipende da:
- L'ordine dei rotori
- I ring settings
- Il plugboard

Ma il plugboard, pur avendo uno spazio enorme (10^{14} configurazioni), influenza S in modo predicibile e separabile!

13.10.5 Il Teorema Fondamentale di Rejewski

Sia S_o la permutazione (posizione 1 → posizione 4) senza plugboard.

Sia π il plugboard.
Allora: $S = \pi \circ S_o \circ \pi^{-1}$

La conseguenza magica: Il plugboard non cambia la struttura ciclica di S (solo "rinomina" le lettere nei cicli).

Esempio:

Se S_o ha il ciclo (A B C D), e il plugboard ha A↔X, B↔Y, allora S avrà il ciclo (X Y C D).

La lunghezza del ciclo (4) rimane invariata!

Rejewski realizzò: La struttura ciclica identifica l'ordine dei rotori + i ring settings, indipendentemente dal plugboard.

13.10.6 Il Catalogo Monumentale

Con questa intuizione, Rejewski progettò il lavoro più ambizioso della crittanalisi pre-computazionale: catalogare la struttura ciclica di S per ogni possibile configurazione di ordine rotori + ring settings.

I parametri:
- L'ordine dei rotori: 6 possibilità (3 da 3)
- I ring settings: $26^3 = 17.576$ per configurazione rotori
- Totale: $6 \times 17.576 = 105.456$ configurazioni

Per ciascuna:
1. Simulare Enigma (con calcoli manuali o macchine additive)
2. Calcolare le permutazioni S, T, U (per le coppie pos 1↔4, 2↔5, 3↔6)
3. Determinare la struttura ciclica (le lunghezze dei cicli)
4. Registrare nel catalogo

Esempio di entry:

```
Configurazione: Rotori I-II-III, Ring A-A-A

S: (4,4,6,3,3,3,2,1) ← lunghezze degli 8 cicli di S
```

T: (3,5,4,6,4,2,2)
U: (5,5,4,4,3,3,2)
```
Lo sforzo computazionale:
Il calcolo manuale per configurazione: ~15 minuti (con macchine additive, tabelle)
Totale: 105.456 × 15 min = 1.581.840 minuti ≈ 3 anni di lavoro continuo!

La soluzione: La parallelizzazione. Un team di ~20 calcolatori (impiegati del Biuro + matematici universitari) lavorarono simultaneamente.

Il completamento: fine 1935 - il catalogo completo stampato su centinaia di pagine.

13.10.7 L'Uso Operativo del Catalogo

La procedura giornaliera:
1. L'intercettazione (mattina): Le stazioni radio polacche intercettano ~60-100 messaggi tedeschi.

2. L'estrazione degli indicatori: Da ogni messaggio, estrarre i 6 caratteri iniziali.

3. La costruzione delle catene: Costruire le permutazioni S, T, U dalle coppie osservate.

Esempio:

```
Messaggio 1: BJGTDN → B→T (posizioni 1↔4), J→D (pos 2↔5), G→N (pos 3↔6)
Messaggio 2: HNMPLQ → H→P, N→L, M→Q
...

Accumulando i dati:

S: B→T, T→K, K→F, F→B (ciclo di lunghezza 4)
  A→E, E→Q, Q→A (ciclo di lunghezza 3)
  ...
```
4. La determinazione della struttura: Calcolare le lunghezze dei cicli di S, T, U.

5. La ricerca nel catalogo: Cercare il match.

```
Struttura osservata:
S: (4,4,3,3,6,6)
```
```

T: (3,5,4,4,5,5)
U: (6,3,3,4,5,5)

Ricerca nel catalogo... → Match!

Configurazione: Rotori II-I-III, Ring 14-22-07 (N-V-G)
```
6. La ricostruzione del plugboard: Con l'ordine dei rotori + i ring settings noti, resta solo il plugboard da determinare (metodo separato, vedi sotto).

7. La decifratura di massa: La configurazione completa applicata a tutti i messaggi del giorno.
Tempo totale con il catalogo: ~2 ore dall'intercettazione alla lettura!

Senza il catalogo: praticamente impossibile.

13.10.8 La Determinazione del Plugboard

Anche con i rotori noti, il plugboard ha ancora ~$10^{11}$ possibilità. Ma Rejewski sviluppò un metodo iterativo intelligente:

L'algoritmo:

1. Ipotesi iniziale: Plugboard vuoto (nessuna connessione).

2. Decifratura tentativa: Alcuni messaggi decifrati con il plugboard vuoto.

3. Ispezione del testo: Parzialmente leggibile, alcune lettere sbagliate.

Esempio: "ANFRIFF UM FUENM UHR" (attacco alle cinque)

Errori evidenti: F↔G (dovrebbe essere "ANGRIFF"), M↔N ("FUENF")

4. La correzione: Aggiungere la coppia G-F al plugboard.

5. La ri-decifratura: Con il nuovo plugboard parziale.

Risultato: "ANGRIFF UM FUENM UHR" - migliorato!

6. L'iterazione: Ripetere fino a ottenere un testo completamente leggibile in tedesco.

Lo sfruttamento dei cribs: I messaggi militari contenevano formule standard:
```

- "OBERKOMMANDO" (comando supremo)
- "WETTERBERICHT" (rapporto meteo)
- "KEINE BESONDEREN EREIGNISSE" (nessun evento particolare - messaggio inviato quando non c'era niente da segnalare!)

Se si sa che il messaggio probabilmente contiene "WETTERBERICHT", si possono fare ipotesi sul plugboard basate sui match parziali.

Tempo medio per il plugboard: 30-60 minuti con esperienza.

13.11 I Successi e i Limiti (1933-1938)

Il periodo d'oro polacco: Dal 1933 al 1938, il Biuro Szyfrów decifrò sistematicamente il traffico Enigma della Wehrmacht.

L'intelligence fornita:
- I movimenti delle truppe tedesche durante l'espansione militare (violazione del Trattato di Versailles)
- I piani strategici
- La struttura organizzativa militare
- Gli sviluppi tattici e dottrinali
- I preparativi per l'annessione dell'Austria (Anschluss, marzo 1938)
- La mobilitazione per la crisi dei Sudeti (settembre 1938)

Le limitazioni:

- Il ritardo: La configurazione veniva determinata dopo ~2 ore. I messaggi erano già "vecchi" - utili per l'intelligence strategica, non per quella tattica in tempo reale.
- La dipendenza dal volume: Servivano molti messaggi (>60) per costruire catene complete. Alcune reti con traffico scarso rimanevano illeggibili.
- Il lavoro manuale intenso: Richiedeva team di crittoanalisti esperti che lavoravano in turni continuati.

13.12 La Crisi del 1938: Il Blackout Totale

15 dicembre 1938: Un giorno nero per il Biuro Szyfrów. Improvvisamente, tutti i metodi smisero di funzionare. Il traffico Enigma divenne completamente illeggibile.
La causa: La Germania introdusse due nuovi rotori (IV e V), aumentando i possibili ordini da 6 a 60.

$P(5,3) = 5 \times 4 \times 3 = 60$

Le conseguenze catastrofiche:

- Il catalogo obsoleto: Copriva solo i rotori I, II, III (6 ordini). Per 60 ordini servirebbero 10× più entries.
- Il tempo impossibile: Un nuovo catalogo completo richiederebbe 10× il tempo originale ≈ 30 anni di lavoro.
- Le bombe insufficienti: Avevano 6 bombe (una per il vecchio ordine). Per 60 ordini servirebbero 60 bombe - un costo proibitivo per il budget polacco.
- La deadline imminente: La guerra era sempre più vicina. La Germania si mobilitava. Non c'erano 30 anni.

Sembrava la fine. Enigma aveva vinto.

13.13 I Fogli di Zygalski: La Genialità sotto Pressione

Henryk Zygalski, il più geometrico dei tre, ebbe un'idea salvifica nel novembre 1938.

Il principio: Sfruttare le "females" - coincidenze speciali che si verificano quando i rotori sono in posizioni particolari.

La female: Quando il rotore medio è sulla sua tacca durante lo stepping, produce un'irregolarità nel meccanismo (il double-stepping). Questa irregolarità lascia una "firma" statistica riconoscibile nei messaggi cifrati.

Il metodo dei fogli perforati:
Per ogni possibile configurazione (ordine rotori + ring settings):

1. Creare un foglio 26×26 (righe = posizione rotore sinistro, colonne = posizione rotore medio)
2. Perforare fori nelle posizioni dove si verificano le females (le coincidenze caratteristiche)

Il set completo: 60 fogli (uno per ogni ordine con 5 rotori)

L'utilizzo:

1. Sovrapporre i fogli corrispondenti alle configurazioni candidate
2. Allineare secondo i dati osservati (le posizioni dove si vedono i patterns nei messaggi intercettati)
3. Cercare le posizioni dove la luce passa attraverso tutti i fogli sovrapposti
4. Le posizioni illuminate = configurazioni possibili (pochissime, testabili manualmente)

La produzione:
La perforazione manuale con precisione millimetrica - un lavoro meticoloso e laborioso.

Il completamento: metà 1939
Con i fogli di Zygalski, i polacchi ripresero parzialmente la capacità di decifratura. Ma la situazione rimaneva precaria, il lavoro era estremamente lento.

13.14 La Conferenza di Pyry: Il Regalo più Prezioso della Storia

25 luglio 1939: Una data che cambiò il corso della guerra.
La Polonia, vedendo la guerra imminente (la Germania aveva già smembrato la Cecoslovacchia a marzo, minacciava la Polonia), prese una decisione strategica audace: condividere tutto con gli alleati.
Il luogo: Pyry, una foresta 20 km a sud di Varsavia. Il centro segreto del Biuro Szyfrów - un edificio anonimo circondato da boschi.

I partecipanti:

Polonia:
- Maggiore Gwido Langer (capo del Biuro)
- Capitano Maksymilian Ciężki (deputy)
- Marian Rejewski
- Jerzy Różycki
- Henryk Zygalski

Francia:
- Capitano Gustave Bertrand (l'uomo che aveva fornito il materiale di Schmidt)
- Maggiore Henri Braquenié

Regno Unito:
- Commander Alastair Denniston (capo del GC&CS - Government Code & Cypher School)
- Dilly Knox (crittoanalista senior, aveva lavorato su Enigma per anni senza successo)
- Commander Humphrey Sandwith
La rivelazione:

Quando Rejewski iniziò a spiegare il metodo delle permutazioni cicliche, la teoria matematica dietro il catalogo, le bombe crittologiche... i britannici rimasero letteralmente a bocca aperta.

Dilly Knox, un matematico di Cambridge e veterano della Room 40 (la crittanalisi navale britannica della WWI), aveva tentato per anni di violare Enigma usando approcci linguistici tradizionali. Un fallimento totale. Ora, questi polacchi - da una nazione considerata marginale militarmente - avevano non solo violato Enigma ma lo avevano fatto con un'eleganza matematica sbalorditiva.
Il materiale consegnato:
- Il metodo completo di Rejewski: La teoria delle permutazioni, gli algoritmi, tutti i dettagli matematici
- Il catalogo: Il set completo (o almeno porzioni significative - le fonti storiche divergono su quanto esattamente fu consegnato fisicamente vs spiegato)
- I progetti delle bombe: Le specifiche complete per la costruzione
- I fogli di Zygalski: Il set completo per il metodo alternativo
- Le repliche di Enigma: Due macchine Enigma militari ricostruite dai polacchi (reverse-engineered). Una per la Francia, una per il Regno Unito.

La reazione britannica:
Denniston tornò a Londra e riferì al governo: "I polacchi ci hanno dato un regalo inestimabile. Abbiamo ora una base per un programma sistematico."
Senza Pyry, Bletchley Park non sarebbe mai esistito nella forma che conosciamo. I britannici avrebbero continuato con approcci empirici probabilmente senza un successo significativo fino a troppo tardi nella guerra.
Settembre 1939: 5 settimane dopo Pyry, la Germania invade la Polonia (1 settembre). La Francia e il Regno Unito dichiarano guerra (3 settembre).
La Seconda Guerra Mondiale inizia. Ma gli alleati hanno ora un'arma segreta: la conoscenza di come violare Enigma.

13.15 L'Esodo Polacco e la Continuità dello Sforzo

1 settembre 1939, ore 04:45: Le corazzate tedesche attraversano il confine polacco. La Luftwaffe bombarda Varsavia. La Polonia, nonostante la resistenza eroica, è schiacciata in 35 giorni dalla Blitzkrieg tedesca e dall'invasione sovietica da est (17 settembre, patto Molotov-Ribbentrop).
Il Biuro Szyfrów aveva preparato piani di evacuazione. Quando divenne chiaro che Varsavia sarebbe caduta, i documenti critici furono distrutti o evacuati, il personale chiave fu disperso.
Settembre 1939: Rejewski, Różycki e Zygalski furono evacuati in Romania attraverso corridoi segreti. Portarono con sé la conoscenza più preziosa del mondo ma quasi nessun documento fisico (troppo rischioso).

Ottobre 1939: Dalla Romania (neutrale ma sotto pressione tedesca), furono trasferiti clandestinamente in Francia via Balcani.

PC Bruno (Francia, 1939-1940): Il Castello di Vignolles, 40 km a nord-est di Parigi. Qui, sotto copertura, i polacchi ripresero il lavoro su Enigma in collaborazione con i francesi (Bertrand) e in collegamento con i britannici (Bletchley Park).

Il setup: Macchine Enigma catturate o ricostruite, fogli di Zygalski, condivisione quotidiana di intelligence con il Regno Unito.

I successi di PC Bruno (inverno 1939-40): La decifratura parziale del traffico della Wehrmacht durante la "Drôle de guerre" (la guerra fasulla - il periodo di relativa calma tra settembre 1939 e maggio 1940).

Giugno 1940: La Germania invade la Francia. Una Blitzkrieg devastante. La Francia crolla in 6 settimane.

14 giugno 1940: I tedeschi entrano a Parigi.

L'evacuazione di PC Bruno: All'ultimo momento. Il team polacco evacua verso sud (la zona non occupata, la Francia di Vichy), portando macchine e documenti critici.

1940-1942: Operazioni clandestine nel sud della Francia. Continuano il lavoro sulle varianti di Enigma (specialmente le versioni italiane e spagnole).

Novembre 1942: La Germania occupa tutto il territorio francese (in risposta allo sbarco alleato in Nord Africa).

13.16 Bletchley Park: La Fabbrica di Ultra

Mentre i polacchi fuggivano attraverso l'Europa in fiamme, in Inghilterra si preparava lo sforzo crittanalitico più massiccio della storia umana.

Bletchley Park: Una mansion vittoriana nel Buckinghamshire, 50 miglia a nord di Londra, acquistata nel 1938 dal Government Code & Cypher School (GC&CS) come sede alternativa in caso di guerra (Londra era vulnerabile ai bombardamenti).

Agosto 1939: Lo staff venne trasferito da Londra a Bletchley (nome in codice "Station X"). Inizialmente c'erano poche dozzine di persone.

1945: Al picco, ~10.000 persone lavoravano a Bletchley e nelle outstations!

L'organizzazione: Un sistema di "huts" (baracche) costruite rapidamente nei terreni:
- Hut 3: Analisi intelligence (Army/Air Force) - traduzione tedesco→inglese, interpretazione, disseminazione

- Hut 4: Analisi intelligence (Naval)
- Hut 6: Crittanalisi Enigma Army/Air Force (capo: Gordon Welchman)
- Hut 8: Crittanalisi Enigma Naval (capo: Alan Turing)
- Hut 11: Le bombe (le macchine crittoanalitiche)
- Altri: Supporto, training, amministrazione

13.17 Alan Turing: Il Genio che Cambiò la Storia

Se Rejewski fu l'architetto della violazione iniziale, Alan Mathison Turing (1912-1954) fu l'ingegnere che la trasformò in un sistema industriale capace di decidere le battaglie.

Settembre 1939: Reclutato dal GC&CS (aveva già lavorato part-time su problemi crittografici), arriva a Bletchley. Viene assegnato alla Hut 8 (Enigma navale) - il problema più critico.

13.18 Il Problema Critico: Enigma Navale e la Battaglia dell'Atlantico

Il contesto strategico: Il Regno Unito è un'isola. La sopravvivenza dipende dalle importazioni via mare:

- Il cibo (il Regno Unito non è autosufficiente)
- Il petrolio
- Le materie prime
- Le munizioni
- Le truppe (dal Commonwealth)

Gli U-Boats (i sottomarini) tedeschi devastavano i convogli atlantici. Gli affondamenti:
- 1940: 1.000+ navi mercantili
- 1941: 1.300+ navi
- 1942: 1.600+ navi (il picco)

Se la Battaglia dell'Atlantico fosse stata persa, il Regno Unito avrebbe dovuto arrendersi per fame - una sconfitta totale.

L'Enigma Naval (Kriegsmarine) era una versione più complessa:

- La procedura diversa: Non usava l'indicatore doppio (già eliminato prima della guerra per la Marina). Usava un sistema più sofisticato con bigram tables.
- I cribs scarsi: I messaggi navali erano brevi, laconici, con poche formule standard (a differenza dell'Army/Air che inviavano rapporti meteorologici prevedibili).

- Il volume limitato: Meno messaggi giornalieri rispetto all'Army - più difficile per l'analisi statistica.

Le conseguenze drammatiche: Ogni giorno di ritardo nella decifratura = navi affondate, marinai morti, rifornimenti persi.

I metodi polacchi (basati sull'indicatore doppio) non funzionavano. Serviva un approccio completamente nuovo.

13.19 Il Metodo del Crib: L'Arma di Turing

Turing sviluppò un metodo basato sul crib - un segmento di testo in chiaro ipotizzato e allineato con il testo cifrato.

Esempio:
Testo cifrato: NQVLT YPBKM UVROM ZJJSI ...
Ipotesi crib: WETTE RVORH ERSAG E____ (previsioni meteo)
└──────────┘
13 caratteri
Le fonti dei cribs:

1. I messaggi stereotipati:
  - "WETTERVORHERSAGE" (previsione meteo) - inviato ogni mattina alle 06:00 dalle stazioni meteo navali
  - "KEINE BESONDEREN EREIGNISSE" (nessun evento particolare)

2. Gli errori operativi:
  - Il re-invio dello stesso messaggio con una chiave diversa (l'operatore si accorge di un errore)
  - Gli operatori pigri (chiavi messaggio prevedibili: AAA, BBB)

3. L'intelligence contestuale:
  - Se si sa che la flotta tedesca è al largo della Norvegia, ci si aspetta nomi geografici nei messaggi

La conseguenza del crib: I Loops

Allineando il crib con il cifrato, creiamo delle corrispondenze:

```
Pos:    1  2  3  4  5  6  7  8  9 10 11 12 13
Chiaro: W  E  T  T  E  R  V  O  R  H  E  R  S
```

Cifrato: N Q V L T Y P B K M U V R
↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕ ↕
```

La proprietà critica di Enigma: Nessuna lettera viene cifrata in se stessa!

Quindi:
- Pos 1: W→N implica N≠W
- Pos 3: T→V implica V≠T
- Pos 13: S→R implica R≠S

Il Loop (l'anello): Una sequenza di connessioni che ritorna al punto iniziale.

Esempio:
- Pos 3: T→V
- Pos 7: V→P (cerchiamo V nel testo in chiaro...)
- ...se il loop si chiude... → P→T
Il loop completo: T→V→P→T

La potenza del loop: Fornisce multiple equazioni simultanee che devono essere soddisfatte dalla configurazione di Enigma. Solo la configurazione corretta soddisfa tutte le equazioni.

13.20 La Bomba di Turing-Welchman

Turing si rese conto che testare manualmente le configurazioni per verificare la consistenza dei loop era possibile ma estremamente lento. Serviva l'automazione.
La Bombe (il nome ereditato dalle bombe polacche, ma con un design completamente diverso) fu il risultato.
La collaborazione tra Alan Turing (la teoria) e Gordon Welchman (l'ingegneria, l'innovazione cruciale del "diagonal board") fu decisiva
Il progetto: Iniziato alla fine del 1939

La prima installazione: 18 marzo 1940, nome "Victory"

Il designer meccanico: Harold "Doc" Keen, British Tabulating Machine Company (BTM)

L'architettura:

Il Drum (tamburo): Una replica elettromeccanica di 3 rotori Enigma. Poteva ruotare attraverso tutte le $26^3$=17.576 posizioni.
```

Il numero di drums per bomba: Tipicamente 36 drums (12 set di 3 rotori ciascuno, corrispondenti a 12 posizioni diverse nel messaggio dove il crib veniva applicato).

Le connessioni: I drums erano interconnessi secondo la struttura del loop identificato nel crib.

Il Diagonal Board (l'innovazione di Welchman): Simulava l'effetto del plugboard, eliminando drammaticamente i falsi positivi. Questa aggiunta aumentò l'efficienza di ~10-50×!

Le specifiche tecniche:

Dimensioni: 200 cm (H) × 150 cm (L) × 60 cm (P)
Peso: ~1000 kg (1 tonnellata)
Drums: 36 rotori fisici (12 set × 3)
Velocità di rotazione: ~120 RPM
Velocità di test: ~2 posizioni/secondo
Tempo per una scansione completa: ~20 minuti (con ottimizzazioni)
Potenza: ~1 kW
Rumore: estremamente rumoroso (clack-clack-clack continuo, da cui il nome "bomba")
```
Il funzionamento:

1. Il setup (operatrici WRNS - "Wrens"):
   - Leggere il "menu" (il diagramma specificante i loop e le connessioni dal crib)
   - Configurare i drums (inserire i rotori nell'ordine specificato per l'ordine testato)
   - Cablare il diagonal board
   - Impostare le posizioni iniziali

2. L'esecuzione:
   - I drums ruotano, testando le posizioni
   - I circuiti elettrici verificano se i loop sono consistenti
   - Contraddizione rilevata → quella posizione viene esclusa (la bomba continua)
   - Nessuna contraddizione → possibile match → i drums si fermano (STOP)

3. L'output:
   - Le posizioni dei rotori candidate vengono registrate dall'operatrice
   - La bomba viene riavviata per cercare altri stop

4. La verifica manuale:
   - Il crittoanalista riceve le posizioni candidate
```

- Le applica a un messaggio di test con una Typex machine (una macchina cifrante britannica configurata come Enigma)
- Se produce tedesco leggibile → SUCCESSO!
- Altrimenti → falso positivo, riavviare la bomba

L'ordine dei rotori: La bomba testava un ordine alla volta. Con 60 possibili ordini (5 rotori scegli 3), per coprire tutto servivano 60 run separati o 60 bombe in parallelo.

La produzione di massa:

Anno	Bombe prodotte	Totale cumulativo
1940	6	6
1941	24	30
1942	49	79
1943	75	154
1944	56	210
1945	0	210 (fine guerra)

Le installazioni: Bletchley Park (Hut 11, poi blocchi dedicati) + outstations (Stanmore, Eastcote, Wavendon)

Il personale: ~2000 WRNS (Women's Royal Naval Service) al picco, operando le bombe in turni 24/7.

13.21 La Routine Operativa: Un Giorno a Bletchley (1942)

03:00 - L'intercettazione: Le stazioni Y (intercettazione radio) sparse per il Regno Unito ricevono il traffico tedesco. Centinaia di messaggi ogni notte.

04:00 - La trasmissione: I messaggi via teleprinter a Bletchley Park.

05:00 - Il sorting (Hut 6/8): Gli analisti ordinano i messaggi per rete (key net). La Wehrmacht usava diverse reti: "Red" (Luftwaffe, Fronte Occidentale), "Light Blue" (Luftwaffe, Mediterraneo), "Dolphin" (U-Boats, Atlantico), ecc.

06:00 - L'identificazione del crib: I crittoanalisti esaminano i messaggi cercando cribs probabili.

Esempio: Messaggio di lunghezza 180 caratteri, ore 06:15, rete "Red" → probabile rapporto meteo → crib "WETTERVORHERSAGE"

06:30 - La costruzione del menu: Disegnare il diagramma dei loop per la bomba.

Esempio di menu:
```
   T(3) ————————— V(7)
    |           |
    |           |
   P(16)—————————+
```

Numeri = posizioni nel messaggio
07:00 - Il setup della bomba: Una Wren assegnata legge il menu, configura i drums, cabla il diagonal board, imposta le posizioni iniziali.

07:15 - Il run della bomba: Avviata. Il rumore assordante riempie la Hut 11.

07:35 - Lo stop: La bomba si ferma - possibile match!

La Wren registra la posizione: Rotori II-V-III, Ring 14-22-07, posizioni P-Q-R

07:37 - Il restart: La bomba viene riavviata per cercare altri stop.

08:00 - La verifica (crittoanalista): Applica la configurazione candidata a un messaggio di test.

Configurazione: II-V-III, ring N-V-G, posizioni P-Q-R, plugboard ancora sconosciuto

08:15 - Il successo!: Con il plugboard parziale (determinato per tentativi), il messaggio produce tedesco:

"WETTERVORHERSAGEVIERZEHNUHRMITTELEUROPAEISCHEZEITWINDNORDOSTSECHS..."

(Previsioni meteo ore 14:00 tempo Europa Centrale vento nord-est 6...)

08:20 - La determinazione del plugboard completo: Metodo iterativo (già descritto). Circa 15-30 minuti.

08:45 - Il break del giorno!: La configurazione completa è determinata.

Annuncio a tutto Bletchley: "RED KEY BROKEN"

09:00-12:00 - La decifratura di massa:

- Setup di ~20 Typex machines configurate come Enigma inverso
- Le operatrici digitano centinaia di messaggi cifrati
- Output: testi tedeschi in chiaro

La traduzione (Hut 3): Un team di linguisti traduce tedesco→inglese

L'analisi intelligence: Identificare le informazioni actionable:
- I movimenti delle truppe
- Gli ordini offensivi/difensivi
- Le disposizioni delle unità
- Le forze aeree disponibili
- La logistica

12:00-14:00 - La disseminazione:

L'intelligence (nome in codice ULTRA) viene inviata via canali sicurissimi a:
- Churchill (copia personale giornaliera)
- Il Comando Supremo Alleato
- I comandanti dei teatri operativi
- L'Ammiragliato (per il routing dei convogli anti-U-Boats)

La Security Ultra (cruciale):

Regola ferrea: Mai agire direttamente sull'intelligence Ultra senza una cover story credibile.

Esempio:
- Ultra rivela la posizione di un U-Boat
- Non si attacca immediatamente
- Si invia un aereo di "ricognizione" nella zona
- L'aereo "avvista" l'U-Boat
- Poi si attacca (cover: spotted by aircraft)

Se i tedeschi avessero sospettato che Enigma fosse compromesso, avrebbero cambiato sistema → la fine di Ultra.

La protezione fanatica: Distribuzione limitatissima. Solo ~300 ufficiali alleati (top brass) con "Ultra clearance". I documenti venivano distrutti dopo la lettura. Zero leaks durante tutta la guerra.

13.22 Il Problema M4: Gli U-Boats e il Blackout del 1942

1 febbraio 1942: Il secondo blackout catastrofico (dopo quello polacco del 1938).

Improvvisamente, il traffico degli U-Boats divenne completamente illeggibile.

La causa: Enigma M4 - una versione a 4-rotori introdotta specificamente per la Flotta sottomarina.

Le differenze dell'M4:
- 4 rotori invece di 3 (il quarto rotore "β" o "γ" non ruotava, sostituendo effettivamente il riflettore standard)
- Un riflettore "sottile" (Thin UKW-B o UKW-C)

L'impatto sullo spazio delle chiavi:

Moltiplicazione per un fattore ~26 (le posizioni extra del rotore)

Le conseguenze strategiche:

Febbraio-dicembre 1942: Il "Secondo periodo felice" per gli U-Boats.

Le perdite dei convogli alleati aumentarono drammaticamente:
- Marzo 1942: 273 navi affondate (534.000 tonnellate)
- Novembre 1942: 119 navi (729.000 tonnellate)

La Battaglia dell'Atlantico era quasi persa. Il Regno Unito era vicino alla fame.

Bletchley era disperata. Le bombe esistenti erano obsolete per l'M4 (progettate per 3 rotori).

30 ottobre 1942: Il Colpo di Fortuna

L'U-559 (Type VIIC) venne affondato dalla HMS Petard al largo di Port Said (Egitto) dopo un lungo inseguimento.

Prima che l'U-Boot affondasse completamente, due marinai britannici - Anthony Fasson (Tenente) e Colin Grazier (Able Seaman) - si tuffarono nel sottomarino in affondamento:

- Scesero nella sala controllo allagata
- Recuperarono i codebooks di Enigma M4
- Recuperarono il Wetterkurzschlüssel (weather short signal book) - cruciale!
- Passarono il materiale a un terzo marinaio sulla superficie

Quando l'U-559 affondò improvvisamente, Fasson e Grazier erano ancora dentro - entrambi morirono annegati.
Il sacrificio eroico: Senza di loro, l'M4 sarebbe rimasto inviolato forse fino alla fine della guerra.

Il materiale recuperato:
- Il Kurzsignalheft (short signal book)
- Il Wetterkurzschlüssel (weather short signals) - un libro con codici meteorologici

Questo libro fu la chiave di tutto: i messaggi meteo degli U-Boats seguivano un formato rigidissimo, fornendo cribs perfetti.

Dicembre 1942: Bletchley decifrò il primo messaggio M4 usando i cribs dai weather reports.

1943: La produzione urgente di bombe "4-rotor":

- Drums modificati per 4 rotori
- Configurazione più complessa
- Velocità ridotta (più posizioni da testare)

Maggio 1943: La piena capacità sull'M4 venne recuperata.

"Black May": Maggio 1943, 43 U-Boats affondati - perdite catastrofiche per la Germania.

Dönitz (il comandante degli U-Boats) ritirò temporaneamente la flotta dall'Atlantico. La Battaglia dell'Atlantico venne effettivamente vinta dagli Alleati.

Fine 1943: Il dominio alleato sull'Atlantico si consolidò. Il Regno Unito era al sicuro.

13.23 L'Impatto di Ultra sulla Seconda Guerra Mondiale

Quantificare l'impatto dell'intelligence su una guerra è notoriamente difficile - le battaglie sono vinte da soldati, marinai e aviatori che combattono e muoiono. Ma Ultra fornì un vantaggio informativo senza precedenti nella storia militare, influenzando ogni teatro di guerra e, secondo alcune stime, accorciando il conflitto di 2-4 anni.

Il Generale Dwight D. Eisenhower (Comandante Supremo Alleato), dopo la guerra:

"Ultra fu decisivo per la vittoria. Senza di esso, la guerra sarebbe durata anni in più, con costi in vite umane quasi incalcolabili. Ogni comandante alleato beneficiò enormemente da questa fonte."
Le stime storiche conservative: 14-21 milioni di vite salvate grazie all'accorciamento della guerra.

13.23.1 La Battaglia d'Inghilterra (Estate 1940)

Dopo la caduta della Francia, la Germania preparò l'invasione del Regno Unito (Operation Sea Lion). Il prerequisito era la supremazia aerea - distruggere la RAF (Royal Air Force).

Luglio-Ottobre 1940: La Luftwaffe attaccò le basi della RAF, le fabbriche e le città britanniche.

Ultra fornì:
- I piani di raid (obiettivi, tempistiche)
- Le forze impegnate (numero di bombardieri, caccia)
- I danni riportati (valutazioni tedesche post-raid)

L'impatto: La RAF poteva allocare i caccia (Spitfires, Hurricanes) ottimalmente, intercettando i raid maggiori, evitando di disperdere le forze su diversivi.

Il risultato: La Germania fallì nell'ottenere la supremazia aerea. L'Operation Sea Lion venne cancellata. Il Regno Unito sopravvisse.

Senza Ultra? La RAF avrebbe comunque combattuto eroicamente, ma con perdite maggiori e forse un esito diverso. Il margine di vittoria fu stretto.

13.23.2 La Battaglia dell'Atlantico (1940-1943)

Già discussa. Ultra permise:

- Il routing evasivo dei convogli: Evitare le posizioni dei wolfpacks di U-Boats note da Ultra
- La caccia attiva: Dirigere le navi di scorta e gli aerei anti-sommergibile verso gli U-Boats identificati
- La valutazione dei danni: Sapere quanti U-Boats erano operativi, dove si rifornivano

Le statistiche:
Periodo pre-Ultra completo (1940-inizio 1942): Perdite alleate 1000+ navi/anno

Periodo post-Ultra M4 (1943-1945): Perdite ridotte del 60-70%. Gli affondamenti di U-Boats aumentarono drammaticamente (781 U-Boats distrutti su 1162 costruiti).

13.23.3 Il Nord Africa (1941-1943)

L'Afrika Korps di Rommel operava in condizioni logistiche precarie - i rifornimenti via mare dall'Italia erano vulnerabili all'interdizione.

Ultra rivelò:
- Le rotte dei convogli di rifornimento
- Lo stato del carburante e delle munizioni di Rommel
- I piani offensivi

L'impatto cruciale - El Alamein (Ottobre-Novembre 1942):

Montgomery (comandante dell'8ª Armata britannica) sapeva da Ultra:

- Rommel aveva carburante per solo 3 giorni di battaglia intensiva
- Le posizioni difensive esatte
- I rinforzi non sarebbero arrivati in tempo

Montgomery pianificò una battaglia di attrito, costringendo Rommel a consumare il carburante. Quando i carri tedeschi rimasero senza, il breakthrough britannico sfondò le linee.

La vittoria decisiva: L'Afrika Korps in ritirata permanente. Il Nord Africa venne liberato entro maggio 1943.

13.23.4 Normandia D-Day (6 Giugno 1944)

Lo sbarco in Normandia - l'operazione più complessa della guerra. Il successo dipendeva dalla sorpresa e dal tenere i rinforzi tedeschi lontano dalle spiagge nelle cruciali prime 48 ore.

Operation Fortitude: Una gigantesca operazione di inganno per convincere Hitler che lo sbarco sarebbe stato a Pas-de-Calais (il punto più stretto della Manica), non in Normandia.

Ultra confermò che l'inganno funzionava:

I messaggi di Enigma intercettati mostravano:
- Hitler era convinto dello sbarco a Calais

- Gli ordini di mantenere le Panzer Divisions a Calais, NON inviarle in Normandia
- Anche dopo il D-Day, Hitler credeva che la Normandia fosse un diversivo

La conseguenza: Gli Alleati stabilirono una testa di ponte in Normandia con un'opposizione ridotta. Quando i tedeschi realizzarono l'errore, era troppo tardi - le forze alleate erano troppo forti.

Senza Ultra? Impossibile sapere se Fortitude sarebbe stato creduto. Forse lo sbarco sarebbe fallito, rimandando la liberazione dell'Europa di anni.

13.23.5 La Battaglia delle Ardenne (Dicembre 1944)

Un paradosso doloroso: Uno dei pochi casi dove Ultra fallì.

L'offensiva tedesca a sorpresa attraverso la foresta delle Ardenne (Belgio). Gli Alleati furono colti completamente impreparati.
Perché Ultra non previde?

- Hitler impose il silenzio radio assoluto per i preparativi
- Gli ordini venivano trasmessi via corriere fisico, non via radio
- C'era pochissimo traffico Enigma intercettabile pre-offensiva

La lezione: Ultra era potentissimo ma non onnisciente. Dipendeva dal fatto che i tedeschi usassero la radio.

Comunque, dopo l'inizio dell'offensiva, il traffico radio riprese. Ultra fornì intelligence critica per la controffensiva alleata che fermò la penetrazione tedesca.

13.23.6 Gli Altri Teatri

Il Fronte Orientale: L'informazione di Ultra venne condivisa selettivamente con l'URSS (Stalin era diffidente, temeva la disinformazione). L'impatto fu limitato.

Il Pacifico: Enigma tedesco era irrilevante. Ma lo sforzo parallelo degli USA sui codici giapponesi (JN-25, Purple) fu ugualmente cruciale - la vittoria di Midway (giugno 1942) fu dovuta all'intelligence sui codici giapponesi.
Il controspionaggio: Ultra rivelò la rete di spie tedesche nel Regno Unito. Quasi tutte vennero catturate, alcune "girate" (double agents nel sistema Double Cross).

13.24 Perché Enigma Fallì: L'Anatomia di una Sconfitta Crittografica

Con la retrospettiva, possiamo identificare vulnerabilità multiple - tecniche, procedurali, psicologiche:

13.24.1 Le Vulnerabilità Tecniche (Inerenti al Design)

1. La proprietà involutoria del riflettore

Nessuna lettera cifrata in se stessa ($E(x) \neq x$).

La conseguenza: Una riduzione drammatica dello spazio di ricerca durante l'attacco con crib. Se il crib ipotizzato ha una E nella posizione dove il cifrato ha una E, quella posizione può essere esclusa immediatamente.

Il miglioramento potenziale: Un riflettore rotante, o l'eliminazione del riflettore (richiederebbe macchine diverse per cifratura e decifratura - una complessità operativa).

2. Lo stepping irregolare (Double-Stepping)

L'anomalia nella meccanica dello stepping creava pattern riconoscibili statisticamente. Zygalski sfruttò questo per i fogli perforati.

Il miglioramento: Una meccanica più precisa che eliminasse il double-step, o uno stepping completamente irregolare (pseudo-random).

3. La limitazione del Plugboard

Solo 10 coppie (20 lettere), 6 lettere unchanged.

Il diagonal board di Welchman ridusse la complessità del plugboard efficacemente.

Il miglioramento: 13 coppie (tutte le lettere) avrebbe eliminato le lettere unchanged, aumentando la complessità.

13.24.2 Le Vulnerabilità Procedurali

1. L'indicatore doppio (Pre-1940)

Cifrare la chiave messaggio due volte (RTZRTZ) creò correlazioni matematiche esatte - il fondamento del metodo di Rejewski.

2. Il Grundstellung standardizzato (Post-1940)

Dopo l'eliminazione dell'indicatore doppio, usare un grundstellung giornaliero uniforme creò nuovi pattern.

Il miglioramento: Un Grundstellung veramente casuale per ogni messaggio, comunicato via un canale separato sicuro.

3. I messaggi stereotipati

"WETTERVORHERSAGE", "KEINE BESONDEREN EREIGNISSE", le formule standard - cribs perfetti.

Il miglioramento: Training degli operatori per variare le formulazioni, evitare la prevedibilità.

4. I re-invii e gli errori

Gli operatori che re-inviavano lo stesso messaggio con una chiave diversa fornivano "depth" (lo stesso testo in chiaro con chiavi diverse) - una vulnerabilità classica.

5. Il volume di traffico

La Wehrmacht usava la radio intensivamente. Più messaggi = più dati per i crittoanalisti.

Il miglioramento: Minimizzare la radio, usare corrieri fisici dove possibile (ma impraticabile per una guerra mobile moderna).

13.24.3 Le Vulnerabilità Psicologiche

1. L'overconfidence

I tedeschi credevano che Enigma fosse matematicamente inviolabile. Anche quando sospettavano una compromissione (es. dopo perdite navali anomale), attribuivano il fatto allo spionaggio tradizionale, non alla violazione crittografica.

Gli indizi ignorati:
- I convogli alleati che evitavano "miracolosamente" i wolfpacks
- Gli attacchi aerei su obiettivi tedeschi con un tempismo sospetto
- La cattura dei documenti dall'U-559 (1942) - avrebbe dovuto allarmare, ma non causò una revisione fondamentale

2. La rigidità operativa

Le procedure standardizzate, cambiate raramente. Facilitava l'addestramento ma creava pattern.

La cultura militare tedesca enfatizzava l'obbedienza agli ordini, non l'iniziativa - gli operatori seguivano le procedure meccanicamente.

3. Le scelte non casuali

Gli operatori umani erano pessimi nello "scegliere casualmente". AAA, QWE, nomi di fidanzate - tutte scorciatoie cognitive sfruttabili.

13.24.4 I Fallimenti dell'Intelligence Tedesca

Il Beobachtungsdienst (B-Dienst - il servizio di intercettazione tedesco) era eccellente, violava i codici navali britannici per parte della guerra. Ma nessuno sforzo equivalente su Enigma stesso. Mancava un "red team" - testare i propri sistemi assumendo un nemico intelligente. Se i tedeschi avessero impiegato matematici brillanti (ne avevano!) per attaccare Enigma come esercizio teorico, avrebbero scoperto le vulnerabilità.

13.25 Le Lezioni per la Crittografia Moderna

Enigma offre lezioni rilevanti ancora oggi:

1. "Complessità ≠ Sicurezza"

Uno spazio delle chiavi enorme (10^{23}) non garantì la sicurezza. Le vulnerabilità strutturali (il riflettore involutorio) superarono la forza bruta.

Moderno: I cifrari moderni (AES, RSA) sono basati su hard problems matematici (teoria dei numeri, algebra), non sulla complessità combinatoria pura.
2. "L'Anello Debole è Sempre Umano"

Le procedure operative, le scelte degli operatori, gli errori umani - fattori più decisivi della matematica.

Moderno: Il security engineering enfatizza la "usability" - i sistemi sicuri devono essere facili da usare correttamente, difficili da usare scorrettamente.

3. "Security Through Obscurity Fails"

Enigma confidava parzialmente nel segreto del metodo. Una volta che il nemico ottenne una macchina (catture, spionaggio), il vantaggio evaporò.

Il Principio di Kerckhoffs (1883, confermato da Enigma): "Il sistema deve essere sicuro anche se il nemico conosce tutto tranne la chiave."

4. "Il Testing Adversariale è Essenziale"

I tedeschi non testarono Enigma contro attaccanti intelligenti e determinati.

Moderno: Le competizioni pubbliche (es. AES competition), i bug bounties, il peer review - assumption: il nemico è brillante e motivato.

5. "La Cripto-Agilità è Critica"

La Germania non poteva cambiare sistema rapidamente. Troppo investimento (100.000 macchine), troppa dipendenza.

Moderno: I sistemi sono progettati per l'algoritmo-agility - la possibilità di switchare algoritmi se viene scoperta una compromissione.

6. "L'Intelligence è un Moltiplicatore di Forza"

Ultra non vinse la guerra da solo - servivano eserciti, flotte, aviazione. Ma permise un uso più efficiente delle risorse, riducendo gli sprechi, aumentando i precision strikes.

Moderno: La cyber intelligence, il SIGINT (signals intelligence) rimangono cruciali per la sicurezza nazionale.

PARTE III: L'ERA DIGITALE E LA CRITTOGRAFIA MODERNA (1945-OGGI)

CAPITOLO 14

LA NASCITA DELLA TEORIA DELL'INFORMAZIONE E LA CRITTOGRAFIA SCIENTIFICA

La fine della Seconda Guerra Mondiale segnò un punto di svolta non solo geopolitico, ma anche intellettuale. Il lavoro segreto condotto durante il conflitto - da Bletchley Park a Los Alamos - aveva dimostrato il potere della matematica applicata su scala industriale. La crittografia, fino ad allora un'arte praticata da pochi specialisti, stava per diventare una disciplina scientifica rigorosa.

14.1 Claude Shannon: Il Padre della Teoria dell'Informazione

Claude Elwood Shannon (1916-2001), un matematico e ingegnere elettrico del MIT, lavorò durante la guerra su problemi di comunicazione sicura per il governo degli Stati

Uniti. Nel 1945, completò un rapporto classificato che sarebbe stato declassificato e pubblicato solo nel 1949 con il titolo "Communication Theory of Secrecy Systems". Questo documento, insieme al suo celebre articolo del 1948 "A Mathematical Theory of Communication", pose le fondamenta matematiche della teoria dell'informazione moderna e della crittografia come scienza.

14.2 Il Concetto di Entropia Informazionale

Shannon introdusse il concetto fondamentale di entropia come misura dell'incertezza o della quantità di informazione.

La definizione formale:

Per una variabile aleatoria discreta X con valori possibili $\{x_1, x_2, ..., x_n\}$ e probabilità $P(x_i)$:

$H(X) = -\Sigma_i P(x_i) \log_2 P(x_i)$

dove H(X) si misura in bit (binary digits).

L'interpretazione:

L'entropia H(X) rappresenta il numero medio di bit necessari per codificare un messaggio dalla sorgente X.

Esempio numerico:

Per una moneta equa (testa o croce con P = 1/2):

$H(X) = -(1/2 \log_2(1/2) + 1/2 \log_2(1/2))$
$= -(1/2 \times (-1) + 1/2 \times (-1))$
$= -(-1/2 - 1/2) = 1$ bit

Un singolo lancio di moneta contiene esattamente 1 bit di informazione.

Per la lingua inglese:

Shannon calcolò che l'entropia della lingua inglese è circa 1,0-1,5 bit per carattere (considerando le correlazioni tra lettere, le parole comuni, la grammatica).

Questo è molto inferiore ai $\log_2(26) \approx 4,7$ bit che servirebbero se ogni lettera fosse equiprobabile e indipendente.

La ridondanza della lingua inglese è quindi circa:

Ridondanza = 1 - (1,5/4,7) ≈ 68%

Questa ridondanza è ciò che rende possibile la crittanalisi basata su pattern linguistici!

14.3 La Perfect Secrecy e il One-Time Pad

Shannon formulò rigorosamente il concetto di perfect secrecy (segretezza perfetta):

Definizione:

Un sistema crittografico ha perfect secrecy se e solo se:

P(M C) = P(M)

per ogni messaggio M e ogni testo cifrato C.

In altre parole: osservare il testo cifrato non fornisce nessuna informazione sul messaggio originale. Il testo cifrato è statisticamente indipendente dal testo in chiaro.

Il Teorema di Shannon (1949):

Un sistema ha perfect secrecy se e solo se:

1. Per ogni coppia (messaggio, testo cifrato) esiste esattamente una chiave che li collega
2. Tutte le chiavi sono equiprobabili
3. La chiave ha entropia almeno pari a quella del messaggio: H(K) ≥ H(M)

Lo One-Time Pad (OTP) - L'unico sistema perfettamente sicuro:

Il principio:

- La chiave K è una sequenza casuale della stessa lunghezza del messaggio M
- La cifratura avviene con XOR bit a bit: C = M ⊕ K
- Ogni chiave viene usata una sola volta (da qui "one-time")
- La chiave è perfettamente casuale e segreta

Esempio:

```
Messaggio (binario):  1 0 1 1 0 1 0 0 1 1 0 1
Chiave (casuale):     0 1 1 0 1 1 0 1 0 1 1 0
                    ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕
Cifrato:              1 1 0 1 1 0 0 1 1 0 1 1
```

La decifratura:

M = C ⊕ K (lo XOR è la sua stessa inversa)

La dimostrazione della perfect secrecy:

Per qualsiasi testo cifrato C osservato, ogni possibile messaggio M è ugualmente probabile perché esiste esattamente una chiave K = M ⊕ C che li collega, e tutte le chiavi sono equiprobabili.

Se C = 1101..., potrebbe essere:
- M = 0000... con K = 1101...
- M = 1111... con K = 0010...
- Qualsiasi altro M con il corrispondente K

Tutte le ipotesi sono ugualmente valide!

Il problema pratico del OTP:
La perfect secrecy ha un costo: la chiave deve essere:

1. Lunga quanto il messaggio - per un libro di 1 MB serve una chiave di 1 MB
2. Perfettamente casuale - difficile da generare (il vero random è costoso)
3. Segreta - deve essere trasmessa in modo sicuro
4. Usata una sola volta - riusare la chiave distrugge la sicurezza

Il problema della distribuzione delle chiavi rende l'OTP impraticabile per la maggior parte delle applicazioni. Se possiamo trasmettere in modo sicuro una chiave lunga quanto il messaggio, potremmo trasmettere direttamente il messaggio!

L'uso pratico limitato:

L'OTP è stato usato storicamente per comunicazioni ad altissima sicurezza:

- La "linea rossa" Washington-Mosca durante la Guerra Fredda
- Comunicazioni diplomatiche critiche

- Sistemi militari strategici dove il costo è giustificato

La chiave veniva distribuita fisicamente via corrieri diplomatici con valigette incatenate ai polsi.

14.4 La Distanza di Unicità

Shannon introdusse il concetto di unicity distance (distanza di unicità):

La definizione:

La lunghezza minima del testo cifrato necessaria affinché esista (con alta probabilità) una sola chiave che produca un testo in chiaro sensato nella lingua target.

La formula:

$U = H(K) / D$

dove:
- $H(K)$ = entropia della chiave (bit)
- D = ridondanza del linguaggio (bit per carattere)

Esempio per il cifrario di sostituzione semplice:
- $H(K) = \log_2(26!) \approx 88{,}4$ bit (numero di permutazioni possibili)
- $D \approx 3{,}2$ bit/carattere (ridondanza dell'inglese)

$U \approx 88{,}4 / 3{,}2 \approx 28$ caratteri

Con circa 28 caratteri di testo cifrato, dovrebbe esistere un'unica decifratura sensata. Nella pratica servono 50-100 caratteri per un attacco affidabile.

L'implicazione:

Anche senza trovare effettivamente la chiave, sappiamo che oltre una certa lunghezza del testo cifrato, l'informazione teorica è sufficiente per una decifratura unica.

14.5 La Confusione e la Diffusione

Shannon identificò due proprietà fondamentali che un buon cifrario dovrebbe avere:

1. La Confusione (Confusion):

Rendere la relazione tra la chiave e il testo cifrato il più complessa possibile.

L'obiettivo: Anche un piccolo cambiamento nella chiave dovrebbe causare un cambiamento drastico e imprevedibile nel testo cifrato.

Come ottenerla: Sostituzioni non lineari, operazioni complesse (le S-box nei cifrari moderni).

2. La Diffusione (Diffusion):

Dissipare la struttura statistica del testo in chiaro attraverso il testo cifrato.

L'obiettivo: Ogni bit del testo in chiaro dovrebbe influenzare molti bit del testo cifrato.

Come ottenerla: Permutazioni, trasposizioni, mixing (le P-box nei cifrari moderni).

Il principio della composizione:

Nessuna singola operazione può fornire sia confusione che diffusione ottimali. La soluzione è alternare operazioni di confusione e diffusione inmultipli "round".

Questo principio guiderà il design di tutti i cifrari a blocchi moderni (DES, AES).

CAPITOLO 15

LA GUERRA FREDDA E LA CRITTOGRAFIA GOVERNATIVA

15.1 La Nascita della NSA

Il 4 novembre 1952, il Presidente Harry Truman firmò un memorandum segreto che creava la National Security Agency (NSA), l'agenzia di intelligence più potente e segreta del mondo.

La missione:

1. SIGINT (Signals Intelligence): Intercettare e decifrare le comunicazioni straniere
2. COMSEC (Communications Security): Proteggere le comunicazioni del governo USA

La NSA ereditò il personale e le competenze da organizzazioni precedenti:

- L'Army Security Agency (ASA)
- La Navy Communications Intelligence Organization

- L'Armed Forces Security Agency (AFSA)

Le dimensioni:

La NSA divenne rapidamente la più grande datrice di lavoro di matematici al mondo:

- Anni '60: ~20.000 dipendenti
- Anni '80: ~40.000 dipendenti
- Oggi: classificato, stimato 30.000-40.000

Il budget: classificato, stimato oggi intorno ai 10 miliardi di dollari/anno.

Il motto: "Defending Our Nation. Securing The Future."
Il soprannome popolare: "No Such Agency" (nessuna agenzia del genere) per l'estrema segretezza.

15.2 I Progetti VENONA e la Crittanalisi Sovietica

Il Progetto VENONA (1943-1980):

Uno degli sforzi di crittanalisi più lunghi e di maggior successo della storia.

Il target: I messaggi cifrati tra Mosca e le ambasciate/consolati sovietici negli USA e in Europa (1940-1948).

Il sistema sovietico: Un sistema di codebook + one-time pad

I sovietici usavano:
1. Un codebook per convertire le parole in gruppi di 4 cifre
2. Un pad di chiavi casuali per sommare (mod 10000) ai codici

Il sistema dovrebbe essere inviolabile (perfect secrecy del OTP).

L'errore fatale:
Per la fretta e le difficoltà logistiche durante la guerra, i sovietici riutilizzarono alcune pagine dei pad!
La riutilizzazione creò "depths" - messaggi diversi cifrati con la stessa chiave.

La tecnica di attacco:

Quando due messaggi M_1 e M_2 sono cifrati con la stessa chiave K:

$C_1 = M_1 + K$

$C_2 = M_2 + K$

Quindi:

$C_1 - C_2 = (M_1 + K) - (M_2 + K) = M_1 - M_2$
La differenza elimina la chiave! Rimane solo la differenza tra i messaggi.

Usando le proprietà statistiche della lingua russa e l'intelligenza contestuale, i crittoanalisti della NSA (e predecessori) ricostruirono progressivamente:

1. Le porzioni di chiave riutilizzate
2. I codebook
3. I messaggi in chiaro

I risultati:
Il progetto VENONA decifrò parzialmente circa 3.000 messaggi (su centinaia di migliaia), ma furono sufficienti per:

- Identificare centinaia di agenti sovietici negli USA e UK
- Confermare che Julius e Ethel Rosenberg erano effettivamente spie sovietiche
- Rivelare l'entità della penetrazione sovietica nel Progetto Manhattan
- Identificare Kim Philby e altri membri del "Cambridge Five"

La lezione:
Anche un sistema teoricamente perfetto (OTP) diventa vulnerabile se le procedure operative non sono seguite rigorosamente. Il riutilizzo delle chiavi fu l'errore fatale.

15.3 La Crittografia Simmetrica nell'Era dei Computer

Con l'avvento dei computer digitali negli anni '50-'60, la crittografia iniziò una transizione fondamentale:

- Da dispositivi elettromeccanici (Enigma, macchine a rotori) a algoritmi software
- Da operazioni su alfabeti (lettere) a operazioni su bit
- Da cifrari speciali per usi specifici a standard universali

I requisiti per la crittografia computerizzata:

1. Velocità: Cifrare/decifrare dati alla velocità della CPU
2. Standardizzazione: Un algoritmo universale per molte applicazioni
3. Sicurezza verificabile: Analisi pubblica e testing
4. Implementazione efficiente: In hardware e software

CAPITOLO 16

IL DATA ENCRYPTION STANDARD (DES)

16.1 La Genesi del DES

All'inizio degli anni '70, il commercio elettronico iniziava a emergere. Le banche avevano bisogno di proteggere le transazioni, le aziende volevano comunicare in modo sicuro. Ma non esisteva uno standard pubblico per la crittografia civile.
Nel 1973, il National Bureau of Standards (NBS), predecessore del NIST (National Institute of Standards and Technology), pubblicò una richiesta per un algoritmo di cifratura standard.

I requisiti:

- Sicurezza crittografica elevata
- Specifica completa e dettagliata
- Disponibile per tutti senza restrizioni
- Adattabile a diverse applicazioni
- Economico da implementare
- Efficiente in uso
- Validabile ed esportabile

La risposta di IBM:

Il team di IBM guidato da Horst Feistel aveva sviluppato una famiglia di cifrari chiamati Lucifer. La versione finale, con alcune modifiche suggerite dalla NSA, fu presentata al NBS.

16.2 L'Architettura di Feistel

Horst Feistel (1915-1990), un crittologo tedesco emigrato negli USA, inventò una struttura geniale che porta il suo nome: la Feistel Network (rete di Feistel).

Il principio:

Un modo elegante di costruire una permutazione invertibile usando una funzione arbitraria (anche non invertibile).

La struttura:

```
Input (64 bit): L₀ R₀ (32 bit sinistri 32 bit destri)

Round i:
   L_{i+1} = R_i
   R_{i+1} = L_i ⊕ F(R_i, K_i)

dove F è una funzione arbitraria, K_i è la sottochiave del round i

Output (dopo n round): R_n L_n
```

La proprietà magica:
La decifratura è identica alla cifratura, ma con le sottochiavi in ordine inverso!

La dimostrazione:

Per invertire un round:

Dato (L_{i+1}, R_{i+1}), vogliamo recuperare (L_i, R_i)

Sappiamo che:
- L_{i+1} = R_i
- R_{i+1} = L_i ⊕ F(R_i, K_i)

Quindi:
- R_i = L_{i+1} (direttamente!)
- L_i = R_{i+1} ⊕ F(R_i, K_i) = R_{i+1} ⊕ F(L_{i+1}, K_i)

Non serve invertire F! Questo è la genialità della costruzione di Feistel.

16.3 La Struttura del DES nel Dettaglio

Il DES (Data Encryption Standard) fu pubblicato come standard federale nel 1977.

I parametri fondamentali:

- Blocco: 64 bit
- Chiave: 56 bit effettivi (64 bit con 8 bit di parità)
- Round: 16
- Struttura: Feistel Network

L'algoritmo completo:

1. La permutazione iniziale (IP):

I 64 bit di input vengono permutati secondo una tabella fissa. Esempio parziale:

IP[1] = 58, IP[2] = 50, IP[3] = 42, ...

Il bit in posizione 1 dell'output viene dal bit in posizione 58 dell'input, ecc.

2. La divisione:
L_0 = bit 1-32 dopo IP
R_0 = bit 33-64 dopo IP

3. I 16 round di Feistel:

Per i = 1 a 16:

```
L_i = R_{i-1}
R_i = L_{i-1} ⊕ F(R_{i-1}, K_i)
```

4. Lo swap finale:

Output = R_{16} L_{16} (nota: l'ordine è invertito!)

5. La permutazione finale (IP^{-1}):

L'inversa della permutazione iniziale.

16.4 La Funzione F: Il Cuore del DES

La funzione F è dove avviene la vera crittografia:

F(R, K) opera su 32 bit di R e 48 bit di sottochiave K:

Step 1: Espansione (E)

I 32 bit di R vengono espansi a 48 bit duplicando alcuni bit secondo una tabella:

```
E(R) = 48 bit
```

Esempio: i bit 1 e 32 vengono duplicati, ecc.
```

Step 2: XOR con la sottochiave

```
E(R) ⊕ K_i = 48 bit
```
Step 3: Le S-box (Substitution boxes)

Il cuore della confusione. I 48 bit vengono divisi in 8 gruppi di 6 bit. Ogni gruppo passa attraverso un'S-box che mappa 6 bit → 4 bit.

L'S-box 1 (esempio):

```
Input: 6 bit = $r_1r_2r_3r_4r_5r_6$

I bit $r_1$ e $r_6$ determinano la riga (0-3)
I bit $r_2r_3r_4r_5$ determinano la colonna (0-15)

La tabella dell'S-box 1:

      Col 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
Riga 0:  14  4 13  1  2 15 11  8  3 10  6 12  5  9  0  7
Riga 1:   0 15  7  4 14  2 13  1 10  6 12 11  9  5  3  8
Riga 2:   4  1 14  8 13  6  2 11 15 12  9  7  3 10  5  0
Riga 3:  15 12  8  2  4  9  1  7  5 11  3 14 10  0  6 13
```

Esempio:

Input: 011011

r_1r_6 = 01 → riga 1
$r_2r_3r_4r_5$ = 1101 = 13 → colonna 13
Output: S_1(011011) = 5 = 0101 (4 bit)

Le S-box sono non lineari - forniscono la confusione di Shannon.

Step 4: La permutazione P
```

I 32 bit risultanti (8 S-box × 4 bit) vengono permutati secondo una tabella fissa - fornisce la diffusione di Shannon.

```
Output F(R, K) = P(S-box output) = 32 bit
```

.

16.5 La Generazione delle Sottochiavi

Dalla chiave principale di 56 bit, vengono derivate 16 sottochiavi K_1, K_2, ..., K_{16} (ciascuna di 48 bit).

Il processo:

1. La PC-1 (Permuted Choice 1):

La chiave di 64 bit (56 + 8 di parità) viene permutata e i bit di parità rimossi → 56 bit

2. La divisione:

C_o = primi 28 bit
D_o = ultimi 28 bit

3. Per ogni round i:

```
C_i = LS_i(C_{i-1})  (left shift)
D_i = LS_i(D_{i-1})
```

dove LS_i è uno shift circolare di 1 o 2 posizioni secondo lo schedule:

```
Round  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
Shift  1  1  2  2  2  2  2  2  1  2  2  2  2  2  2  1
```

4. La PC-2 (Permuted Choice 2):

```
K_i = PC-2(C_i  D_i)

Seleziona 48 bit dai 56 bit di C_i  D_i
```

16.6 La Controversia sulla Sicurezza del DES

Quando il DES fu proposto, scoppiò una controversia nella comunità scientifica:

1. La lunghezza della chiave (56 bit):

La critica: 56 bit sono troppo pochi! $2^{56} \approx 7{,}2 \times 10^{16}$ chiavi sono vulnerabili alla forza bruta.
IBM aveva proposto inizialmente 128 bit, ma la NSA suggerì di ridurre a 56 bit.
Il sospetto: La NSA voleva poter rompere il DES con i suoi supercomputer, mantenendo fuori altri attori.
La conferma (1998): La EFF (Electronic Frontier Foundation) costruì "Deep Crack", una macchina specializzata che trovava una chiave DES in media in 4,5 giorni, al costo di $250.000.
Oggi: Con i computer moderni, il DES si rompe in ore/minuti.

2. Le S-box misteriose:

Le S-box del DES sembravano generate arbitrariamente. La NSA non spiegò i criteri di design.

Il sospetto: Backdoor intenzionali? Debolezze note solo alla NSA?

La rivelazione (anni '90): I criteri di design delle S-box furono declassificati. Erano progettate per resistere alla crittanalisi differenziale, una tecnica scoperta pubblicamente solo negli anni '90 da Biham e Shamir!
La NSA conosceva la crittanalisi differenziale già negli anni '70 (!) e aveva fortificato il DES contro di essa.

Il verdetto finale:

Il DES è un algoritmo solido, senza backdoor note. La vera debolezza è solo la lunghezza della chiave, inadeguata per gli standard moderni.

16.7 Il Triple DES (3DES)

Per estendere la vita del DES, fu sviluppato il Triple DES:

L'idea: Applicare il DES tre volte con chiavi diverse.

La variante standard (3DES-EDE):

C = E_{K3}(D_{K2}(E_{K1}(M)))

dove E_K = cifratura DES con chiave K
D_K = decifratura DES con chiave K
```
Perché Encrypt-Decrypt-Encrypt? Per compatibilità: se $K_1 = K_2 = K_3$, allora 3DES = DES singolo.
La lunghezza effettiva della chiave:

- 3 chiavi × 56 bit = 168 bit (teorico)
- Ma attacchi meet-in-the-middle riducono la sicurezza a circa 112 bit
- Comunque molto meglio dei 56 bit originali!

L'uso:

3DES è stato lo standard de facto per le applicazioni finanziarie (ATM, carte di credito) per decenni.

Lo status attuale:
Deprecato dal NIST nel 2017, da eliminare entro 2023. Sostituito dall'AES.

CAPITOLO 17

LA RIVOLUZIONE DELLA CRITTOGRAFIA A CHIAVE PUBBLICA

17.1 Il Problema della Distribuzione delle Chiavi

Tutti i sistemi crittografici fino agli anni '70 erano simmetrici: la stessa chiave usata per cifrare e decifrare.

Il problema fondamentale:
Come fanno Alice e Bob a concordare una chiave segreta se:
- Non si sono mai incontrati di persona?
- Tutti i canali di comunicazione sono insicuri (intercettabili)?

Le soluzioni tradizionali:

1. Corrieri fisici: Costoso, lento, vulnerabile
2. Rete di distribuzione delle chiavi: Complessa, richiede infrastruttura
3. Centro di distribuzione delle chiavi (KDC): Punto singolo di fallimento

Per n utenti che vogliono comunicare a coppie:
- Chiavi necessarie: n(n-1)/2
```

- Per n = 1.000.000: circa 500 miliardi di chiavi!

Il problema diventa intrattabile su larga scala.

17.2 L'Idea Rivoluzionaria: Whitfield Diffie e Martin Hellman

Nel 1976, due ricercatori della Stanford University - Whitfield Diffie e Martin Hellman - pubblicarono un articolo che avrebbe cambiato per sempre la crittografia ("New Directions in Cryptography").
L'idea centrale era radicale: è possibile concordare una chiave segreta su un canale pubblico non sicuro!

17.3 Lo Scambio di Chiavi Diffie-Hellman

Il protocollo:

Alice e Bob vogliono concordare una chiave segreta. Tutti i messaggi sono intercettabili.

Setup pubblico:

- p = un numero primo grande (es. 2048 bit)
- g = un generatore del gruppo moltiplicativo mod p

Questi parametri sono pubblici, noti a tutti incluso l'attaccante Eva.

Il protocollo:

1. Alice:
- Sceglie un numero segreto a (es. casuale tra 1 e p-1)
- Calcola A = g^a mod p
- Invia A a Bob (pubblicamente)

2. Bob:
- Sceglie un numero segreto b
- Calcola B = g^b mod p
- Invia B ad Alice (pubblicamente)

3. Alice e Bob calcolano la chiave condivisa:

Alice calcola: s = B^a mod p = (g^b)^a mod p = g^{ab} mod p
Bob calcola: s = A^b mod p = (g^a)^b mod p = g^{ab} mod p

Hanno entrambi lo stesso valore s = g^{ab} mod p!

Il fatto sorprendente:
Eva (l'attaccante) vede:
- p, g (pubblici)
- A = g^a mod p
- B = g^b mod p

Ma non può calcolare s = g^{ab} mod p senza conoscere a o b!

Il Problema del Logaritmo Discreto (DLP):

Dato g, p, e A = g^a mod p, trovare a.

Questo problema è considerato computazionalmente intrattabile per p sufficientemente grande (2048+ bit).

Esempio numerico (con numeri piccoli per illustrazione):

```
p = 23, g = 5

Alice sceglie a = 6
A = 5^6 mod 23 = 15625 mod 23 = 8
Alice invia A = 8

Bob sceglie b = 15
B = 5^15 mod 23 = 30517578125 mod 23 = 19
Bob invia B = 19

Alice calcola: s = 19^6 mod 23 = 47045881 mod 23 = 2
Bob calcola: s = 8^15 mod 23 = 35184372088832 mod 23 = 2

Chiave condivisa: s = 2 ✓
```
Eva vede i valori pubblici (23, 5, 8, 19) ma non può facilmente ricavare la chiave condivisa 2 senza conoscere i segreti a o b.

Il vantaggio fondamentale:

Due parti che non si sono mai incontrate possono stabilire una chiave segreta condivisa attraverso un canale completamente pubblico e intercettabile. Questo risolve il problema millenario della distribuzione delle chiavi.

La limitazione:
Diffie-Hellman non fornisce autenticazione. Eva potrebbe effettuare un attacco man-in-the-middle:
- Intercettare il messaggio di Alice e sostituirlo
- Stabilire una chiave con Alice e una diversa con Bob
- Decifrare, leggere e ri-cifrare tutti i messaggi

Per questo motivo, Diffie-Hellman deve essere combinato con un meccanismo di autenticazione (certificati, firme digitali).

17.4 L'RSA: La Prima Crittografia a Chiave Pubblica Completa

Diffie-Hellman risolse lo scambio di chiavi, ma non permetteva la cifratura diretta di messaggi. Nel 1977, tre ricercatori del MIT - Ronald Rivest, Adi Shamir e Leonard Adleman - inventarono il sistema RSA, il primo sistema completo di crittografia a chiave pubblica.

Il concetto rivoluzionario:

Invece di una chiave simmetrica condivisa, ogni utente possiede:
- Una chiave pubblica (distribuibile liberamente) - per cifrare messaggi destinati a lui
- Una chiave privata (mantenuta segreta) - per decifrare i messaggi ricevuti

La matematica dell'RSA:

1. La generazione delle chiavi:

- Scegliere due numeri primi grandi p e q (tipicamente 1024 bit ciascuno)
- Calcolare $n = p \times q$ (il modulo RSA, tipicamente 2048 bit)
- Calcolare $\phi(n) = (p-1)(q-1)$ (la funzione di Eulero)
- Scegliere un esponente pubblico e (comunemente $e = 65537 = 2^{16} + 1$)
- Calcolare l'esponente privato d tale che: $d \times e \equiv 1 \pmod{\phi(n)}$

La chiave pubblica: (n, e)
La chiave privata: (n, d)

2. La cifratura:

Per cifrare un messaggio M (dove $0 \leq M < n$):

$C = M^e \bmod n$

3. La decifratura:

Per decifrare il testo cifrato C:
$M = C^d \bmod n$

La dimostrazione della correttezza:

Vogliamo dimostrare che $(M^e)^d \equiv M \pmod n$

Per il teorema di Eulero: se MCD(M, n) = 1, allora $M^{\phi(n)} \equiv 1 \pmod n$

Sappiamo che $d \times e \equiv 1 \pmod{\phi(n)}$, quindi $d \times e = 1 + k \times \phi(n)$ per qualche intero k

Quindi:
$(M^e)^d = M^{(ed)} = M^{(1+k\times\phi(n))} = M \times (M^{\phi(n)})^k \equiv M \times 1^k \equiv M \pmod n$ ✓

Esempio numerico (con numeri piccoli):

```
1. Generazione delle chiavi:
   p = 61, q = 53
   n = 61 × 53 = 3233
   ϕ(n) = 60 × 52 = 3120
   e = 17 (scelto coprimo con ϕ(n))

   Calcolare d: 17 × d ≡ 1 (mod 3120)
   d = 2753

   Chiave pubblica: (3233, 17)
   Chiave privata: (3233, 2753)

2. Cifratura:
   Messaggio: M = 123
   C = 123^17 mod 3233 = 855

3. Decifratura:
   M = 855^2753 mod 3233 = 123 ✓
```

La sicurezza dell'RSA:

Si basa sulla difficoltà di fattorizzare n in p e q. Se un attaccante potesse fattorizzare n, potrebbe calcolare $\phi(n) = (p-1)(q-1)$ e quindi ricavare d dalla relazione $d \times e \equiv 1 \pmod{\phi(n)}$.

Il Problema della Fattorizzazione:

Dato $n = p \times q$, trovare p e q. Nessun algoritmo efficiente è noto per computer classici. Il miglior algoritmo (General Number Field Sieve) ha complessità sub-esponenziale ma comunque intrattabile per n sufficientemente grande.

Le dimensioni delle chiavi consigliate:

- Anni '90: 1024 bit (oggi considerato insicuro)
- Anni 2000-presente: 2048 bit (standard attuale)
- Per sicurezza a lungo termine: 4096 bit

I record di fattorizzazione:

- 2009: RSA-768 (768 bit) fattorizzato
- 2020: RSA-250 (829 bit) fattorizzato
- RSA-2048 rimane sicuro (si è stimato che può richiedere miliardi di anni con la tecnologia attuale)

17.5 Le Firme Digitali con RSA

Una proprietà straordinaria dell'RSA: le operazioni sono commutative. Questo permette di implementare firme digitali.

Il meccanismo della firma:

Alice vuole firmare un messaggio M:

1. Alice calcola la firma: $S = M^{(d_a)} \bmod n_a$ (usando la sua chiave privata!)
2. Alice invia (M, S)
3. Bob verifica: calcola $M' = S^{(e_a)} \bmod n_a$ (usando la chiave pubblica di Alice)
4. Se M' = M, la firma è valida ✓

Perché funziona:

$M' = S^{(e_a)} = (M^{(d_a)})^{(e_a)} = M^{(d_a \times e_a)} = M^{(e_a \times d_a)} = (M^{(e_a)})^{(d_a)} = M$

Solo Alice, possedendo d_a, può produrre S tale che $S^{(e_a)} = M$.
Le proprietà garantite:
1. Autenticità: Solo Alice può creare la firma
2. Integrità: Qualsiasi modifica di M invalida la firma
3. Non ripudio: Alice non può negare di aver firmato (la sua chiave privata è coinvolta)

L'implementazione pratica:

Nella pratica, si firma l'hash del messaggio (non il messaggio diretto) per efficienza:

$S = H(M)^{(d_a)} \bmod n_a$

dove H è una funzione di hash crittografica (SHA-256, ecc.). Questo permette di firmare messaggi di qualsiasi lunghezza producendo sempre firme di dimensione fissa.

17.6 L'Impatto della Crittografia a Chiave Pubblica

La scoperta della crittografia a chiave pubblica rivoluzionò completamente il campo della sicurezza informatica.

Prima (solo crittografia simmetrica):
- n utenti → $O(n^2)$ chiavi da gestire e distribuire
- Richiede canale sicuro per scambio chiavi
- Nessuna firma digitale possibile
- Infrastruttura di gestione chiavi estremamente complessa

Dopo (crittografia asimmetrica):
- n utenti → n coppie di chiavi (pubblica/privata)
- Nessun canale sicuro necessario per distribuzione chiavi pubbliche
- Firme digitali native nel sistema
- Infrastruttura notevolmente semplificata

Le applicazioni moderne fondamentali:

1. SSL/TLS (HTTPS): Le connessioni web sicure usano RSA o ECDH per lo scambio di chiavi, poi cifratura simmetrica per i dati
2. PGP/GPG: Email cifrate e firmate digitalmente
3. SSH: Accesso remoto sicuro ai server
4. Bitcoin e blockchain: Le firme digitali garantiscono la proprietà e l'autenticità delle transazioni

5. Certificati digitali: PKI (Public Key Infrastructure) per l'autenticazione di siti web e servizi

La rivelazione del GCHQ:
Nel 1997 fu rivelato che il GCHQ britannico (Government Communications Headquarters) aveva scoperto indipendentemente i concetti della crittografia a chiave pubblica alcuni anni prima di Diffie, Hellman e RSA:

- James Ellis (1970): Concetto generale di crittografia a chiave pubblica
- Clifford Cocks (1973): Algoritmo equivalente all'RSA
- Malcolm Williamson (1974): Scambio di chiavi simile a Diffie-Hellman

Tuttavia, questo lavoro rimase classificato. Furono Diffie, Hellman, Rivest, Shamir e Adleman a pubblicare pubblicamente e rendere disponibile questa rivoluzione crittografica al mondo intero.

CAPITOLO 18

L'ADVANCED ENCRYPTION STANDARD (AES)

18.1 La Necessità di un Successore del DES

Alla fine degli anni '90, divenne chiaro che il DES aveva raggiunto la fine della sua vita utile:

- La chiave da 56 bit era troppo corta (vulnerabile alla forza bruta)
- Il 3DES era lento (tre volte le operazioni del DES)
- Serviva un algoritmo moderno, efficiente e sicuro per il XXI secolo

Nel 1997, il NIST (National Institute of Standards and Technology) lanciò un concorso pubblico internazionale per selezionare il nuovo Advanced Encryption Standard (AES).

I requisiti richiesti:

1. Dimensioni del blocco: 128 bit
2. Dimensioni delle chiavi: 128, 192, 256 bit
3. Sicurezza: Resistente a tutti gli attacchi noti
4. Efficienza: Veloce sia in software che in hardware
5. Flessibilità: Implementabile su diverse piattaforme (PC, smartcard, dispositivi embedded)
6. Design: Semplice, elegante, analizzabile pubblicamente

Il processo di selezione:

1. 1998: 15 algoritmi candidati presentati da team internazionali
2. 1999: Prima conferenza AES - analisi pubblica e crittoanalisi
3. 2000: 5 finalisti selezionati: Rijndael, Serpent, Twofish, RC6, MARS
4. 2001: Rijndael selezionato come vincitore
5. 2002: AES pubblicato come standard federale (FIPS 197)

18.2 Rijndael: Il Vincitore

Gli inventori: Joan Daemen e Vincent Rijmen, crittografi belgi.
Il nome: Rijndael = combinazione dei cognomi: Rijmen + Daemen

Perché vinse Rijndael:

- Eccellente equilibrio sicurezza/prestazioni
- Design elegante basato su principi matematici solidi
- Molto efficiente sia in software che in hardware
- Resistente a tutti gli attacchi noti
- Struttura semplice e comprensibile
- Margin di sicurezza ampio

18.3 La Struttura dell'AES

A differenza del DES (rete di Feistel), l'AES usa una rete di sostituzione-permutazione (SP network).

I parametri base:

- Blocco: 128 bit (16 byte)
- Chiave: 128, 192, o 256 bit
- Round:
  - AES-128: 10 round
  - AES-192: 12 round
  - AES-256: 14 round

La rappresentazione dello stato:

I 16 byte del blocco sono organizzati in una matrice 4×4 (lo "state"):

```
```

S_0 S_4 S_8 S_{12}
S_1 S_5 S_9 S_{13}

S_2 S_6 S_{10} S_{14}
S_3 S_7 S_{11} S_{15}
```

L'AES opera su questa matrice, trasformandola attraverso i round.

## 18.4 Le Quattro Operazioni Fondamentali

Ogni round dell'AES (eccetto l'ultimo) applica quattro trasformazioni in sequenza:

### 1. SubBytes (Sostituzione non lineare)

Ogni byte dello state viene sostituito usando un'S-box basata sull'inverso moltiplicativo nel campo di Galois $GF(2^8)$.

La costruzione dell'S-box:

Per ogni byte b (da 0x00 a 0xFF):

1. Calcolare l'inverso moltiplicativo in $GF(2^8)$: $b^{-1}$ (Convenzione: $0^{-1} = 0$)
2. Applicare una trasformazione affine che introduce non-linearità aggiuntiva

L'S-box completa è pre-calcolata e memorizzata come tabella di lookup (256 valori).

Esempio:
- Input: 0x53
- Output: 0xED (dopo inverso moltiplicativo e trasformazione affine)

Proprietà dell'S-box dell'AES:

- Altamente non lineare (essenziale per resistere agli attacchi)
- Nessun punto fisso ($S[a] \neq a$ per ogni $a \neq 0$)
- Nessun punto opposto ($S[a] \oplus S[a \oplus FF] \neq FF$)
- Resistente alla crittanalisi differenziale e lineare

### 2. ShiftRows (Permutazione delle righe)

Le righe dello state vengono ruotate ciclicamente a sinistra:
```
Riga 0: nessuna rotazione
Riga 1: rotazione di 1 posizione
Riga 2: rotazione di 2 posizioni
Riga 3: rotazione di 3 posizioni
```
```

Prima: Dopo:

S_0 S_4 S_8 S_{12} S_0 S_4 S_8 S_{12}
S_1 S_5 S_9 S_{13} → S_5 S_9 S_{13} S_1
S_2 S_6 S_{10} S_{14} S_{10} S_{14} S_2 S_6
S_3 S_7 S_{11} S_{15} S_{15} S_3 S_7 S_{11}

```
```

Questo assicura che i byte di una colonna vengano distribuiti su colonne diverse, aumentando la diffusione.

3. MixColumns (Mixing delle colonne)

Ogni colonna dello state viene moltiplicata per una matrice fissa in GF(2^8):

```
```

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (\text{in } GF(2^8))$$

```
```

I coefficienti (02, 03, 01) rappresentano polinomi in GF(2^8):
- 01 = identità
- 02 = moltiplicazione per x
- 03 = moltiplicazione per x+1

Esempio di calcolo:

$$s'_0 = (02 \bullet s_0) \oplus (03 \bullet s_1) \oplus (01 \bullet s_2) \oplus (01 \bullet s_3)$$

dove • indica la moltiplicazione in GF(2^8) con polinomio irriducibile $p(x) = x^8 + x^4 + x^3 + x + 1$.

L'effetto: Ogni byte di output dipende da tutti i 4 byte della colonna di input - diffusione massima all'interno di ogni colonna.

4. AddRoundKey (XOR con la sottochiave)
Lo state viene XORato bit a bit con la sottochiave del round:

$$State \leftarrow State \oplus RoundKey_i$$

Questa è l'unica operazione che dipende dalla chiave segreta. È la parte che introduce la "confusione" dipendente dalla chiave.

18.5 Lo Schedule delle Chiavi (Key Expansion)

Dalla chiave principale K (128/192/256 bit), vengono derivate le sottochiavi per tutti i round.

Per AES-128 (chiave di 128 bit = 16 byte = 4 word):

Le 4 word iniziali W[0], W[1], W[2], W[3] sono la chiave originale.

Per i = 4 a 43 (servono 44 word per 11 round-keys):

```
Se i mod 4 = 0:
   temp = SubWord(RotWord(W[i-1])) ⊕ Rcon[i/4]
Altrimenti:
   temp = W[i-1]

W[i] = W[i-4] ⊕ temp
```

Dove:
- RotWord: rotazione circolare dei byte: $[a_0,a_1,a_2,a_3] \rightarrow [a_1,a_2,a_3,a_0]$
- SubWord: applicazione dell'S-box a ogni byte
- Rcon[i]: costante del round = [x^(i-1), 0, 0, 0] in $GF(2^8)$

Esempio di Rcon:

```
Rcon[1] = [01, 00, 00, 00]
Rcon[2] = [02, 00, 00, 00]
Rcon[3] = [04, 00, 00, 00]
Rcon[4] = [08, 00, 00, 00]
...
Rcon[10] = [36, 00, 00, 00]
```

18.6 L'Algoritmo Completo dell'AES-128

Cifratura:

```
Input: plaintext P (128 bit), chiave K (128 bit)
Output: ciphertext C (128 bit)
```

```
1. KeyExpansion(K) → RoundKey[0..10]
2. State ← P (organizzato in matrice 4×4)
3. AddRoundKey(State, RoundKey[0])

4. Per round = 1 a 9:
   SubBytes(State)
   ShiftRows(State)
   MixColumns(State)
   AddRoundKey(State, RoundKey[round])

5. Round finale (senza MixColumns):
   SubBytes(State)
   ShiftRows(State)
   AddRoundKey(State, RoundKey[10])

6. C ← State
   Return C
```

Decifratura:

La decifratura applica le trasformazioni inverse in ordine inverso:

- InvSubBytes (usando l'S-box inversa)
- InvShiftRows (rotazione a destra invece che a sinistra)
- InvMixColumns (moltiplicazione con la matrice inversa)
- AddRoundKey (lo XOR è la sua stessa inversa)

18.7 Le Proprietà di Sicurezza dell'AES

1. Resistenza alla crittanalisi differenziale:

L'S-box e MixColumns sono progettati per limitare la probabilità di caratteristiche differenziali utilizzabili in attacchi.

2. Resistenza alla crittanalisi lineare:

La non-linearità dell'S-box rende improbabili le approssimazioni lineari efficaci.

3. Il wide-trail design:

Dopo solo 2 round, ogni byte di output dipende da tutti i byte di input - diffusione rapida e completa. Questo è un principio di design fondamentale che garantisce la robustezza.
4. Nessun punto debole strutturale:

A differenza del DES, non ci sono round-key deboli o chiavi semi-deboli note.

Gli attacchi noti (tutti impraticabili nella pratica):

- AES-128: Attacco con complessità 2^{126} (marginalmente meglio della forza bruta 2^{128})
- AES-256: Attacco con complessità 2^{254} (vs forza bruta 2^{256})

Tutti gli attacchi conosciuti sono puramente teorici - richiedono risorse astronomiche impossibili da realizzare in pratica.

18.8 Le Prestazioni dell'AES

In software (CPU Intel Core i7, single-core):

- AES-128: circa 10-15 cicli/byte (circa 1-2 GB/s)
- Con AES-NI (istruzioni hardware): circa 0,6 cicli/byte (circa 20 GB/s)

In hardware (FPGA/ASIC):

- Throughput: 10-100 Gbps con implementazioni parallele

Il confronto con i predecessori:

Algoritmo	Velocità (SW)	Sicurezza	Note
DES	Media	Rotta	56 bit troppo pochi
3DES	Lenta (1/3 DES)	Adeguata	Deprecato
AES-128	Veloce	Eccellente	Standard attuale
AES-256	Veloce	Eccellente	Massima sicurezza

L'adozione universale:
L'AES è oggi lo standard de facto per la crittografia simmetrica in quasi ogni applicazione:

- Protocolli: TLS/SSL, IPsec, SSH, WPA2/WPA3
- File system: BitLocker, FileVault, dm-crypt
- Database: MySQL, PostgreSQL, Oracle
- Cloud storage: AWS, Azure, Google Cloud

- Messaggistica: Signal, WhatsApp, Telegram

CAPITOLO 19

LE FUNZIONI DI HASH CRITTOGRAFICHE

19.1 Il Concetto di Funzione di Hash

Una funzione di hash crittografica è una funzione che mappa dati di lunghezza arbitraria in una stringa di lunghezza fissa (il "digest" o "hash").

La definizione formale:

$H : \{0,1\} \rightarrow \{0,1\}^n$

Dove:
- {0,1} = tutte le stringhe binarie di qualsiasi lunghezza
- $\{0,1\}^n$ = stringhe binarie di lunghezza fissa n

Esempio:

SHA-256 mappa qualsiasi input (file, messaggio, ecc.) in un output di esattamente 256 bit.

19.2 Le Proprietà Richieste

Una funzione di hash crittografica deve soddisfare tre proprietà fondamentali:

1. Resistenza alla pre-immagine (One-way property):

Dato h = H(m), deve essere computazionalmente intrattabile trovare qualsiasi m' tale che H(m') = h.

Complessità attesa: $O(2^n)$ per un hash di n bit.

L'analogia: Un tritacarne - facile andare avanti (carne → macinato), impossibile tornare indietro (macinato → carne originale).

2. Resistenza alla seconda pre-immagine (Weak collision resistance):

Dato m_1, deve essere computazionalmente intrattabile trovare $m_2 \neq m_1$ tale che $H(m_1) = H(m_2)$.

Complessità attesa: $O(2^n)$
3. Resistenza alle collisioni (Strong collision resistance):

Deve essere computazionalmente intrattabile trovare qualsiasi coppia (m_1, m_2) con $m_1 \neq m_2$ tale che $H(m_1) = H(m_2)$.

Complessità attesa: O(2^(n/2)) per il paradosso del compleanno.

19.3 Il Paradosso del Compleanno

Il problema classico:

Quante persone servono in una stanza affinché la probabilità che due condividano lo stesso compleanno superi il 50%?

La risposta contro-intuitiva: Solo 23 persone!

La generalizzazione per le funzioni di hash:

Per trovare una collisione in uno spazio di 2^n valori possibili, servono circa $\sqrt{(2^n)} = 2^{(n/2)}$ tentativi.

L'implicazione per la sicurezza:

- MD5 (128 bit): collisioni trovabili con circa 2^{64} operazioni (fattibile!)
- SHA-1 (160 bit): collisioni con circa 2^{80} operazioni (al limite)
- SHA-256 (256 bit): collisioni con circa 2^{128} operazioni (sicuro)

La dimostrazione (semplificata):

Probabilità che n persone abbiano compleanni tutti diversi:
P(no collision) = (365/365) × (364/365) × (363/365) × ... × ((365-n+1)/365)

Per n = 23: P(no collision) ≈ 0,493, quindi P(collision) ≈ 0,507 > 50%

Per funzioni di hash con 2^n valori possibili:
Dopo $\sqrt{(2^n)} = 2^{(n/2)}$ tentativi, P(collision) ≈ 50%

19.4 Le Applicazioni delle Funzioni di Hash

1. L'integrità dei dati:

File originale → H(file) = hash_originale
File ricevuto → H(file) = hash_ricevuto

Se hash_originale = hash_ricevuto → file integro ✓
```
Usato in: download di software, blockchain, sistemi di backup.

2. Le firme digitali:

Invece di firmare direttamente un messaggio lungo M, si firma il suo hash:
```
Firma = Sign(H(M), chiave_privata)
```

Vantaggi:
- Velocità (l'hash è fisso e piccolo, es. 256 bit)
- Sicurezza (l'hash cattura l'intero contenuto di M)

3. La memorizzazione delle password:

MAI memorizzare password in chiaro!

Database: memorizza H(password)

Al login:
- Utente invia password
- Server calcola H(password)
- Confronta con il valore memorizzato
```
Anche se il database viene rubato, l'attaccante non può recuperare le password originali (resistenza alla pre-immagine).

Il miglioramento: Salt + Hashing:

salt = valore casuale unico per utente
Memorizzare: (salt, H(password salt))
Il salt previene gli attacchi con rainbow tables (tabelle pre-calcolate di hash comuni).

4. Le Proof-of-Work (Bitcoin):

Trovare un nonce tale che:

H(header_blocco nonce) < target

dove target è un numero molto piccolo (molti zero iniziali richiesti).

Il mining di Bitcoin consiste nel provare trilioni di nonce fino a trovarne uno che soddisfi questa condizione.

5. Le HMAC (Hash-based Message Authentication Code):

Combinare hashing con una chiave segreta per autenticare messaggi:

```
HMAC(K, M) = H((K ⊕ opad)  H((K ⊕ ipad)  M))

dove:
- opad = 0x5C ripetuto
- ipad = 0x36 ripetuto
```

Fornisce sia integrità che autenticazione.

19.5 La Famiglia MD (Message Digest)

MD5 (1991) - Ronald Rivest:
- Output: 128 bit
- Struttura: Merkle-Damgård con 64 round
- Status: ROTTO - collisioni trovabili in secondi (dal 2004)

L'attacco a MD5:

Nel 2004, Wang et al. trovarono un metodo per generare collisioni MD5 in poche ore.
Nel 2008, furono creati due certificati SSL diversi con lo stesso hash MD5 - un disastro per la sicurezza!
Uso attuale: Solo per checksum non critici. MAI per crittografia.

19.6 La Famiglia SHA (Secure Hash Algorithm)

SHA-1 (1995) - NSA:

- Output: 160 bit
- Struttura: Merkle-Damgård, basato su MD5 ma rinforzato

- Status: DEPRECATO - collisioni teoriche (2005), pratiche (2017)

L'attacco SHAttered (2017):

Google produsse la prima collisione SHA-1 pratica:
- Due PDF diversi con lo stesso SHA-1
- Costo: circa 110 anni GPU equivalenti
- Dimostrazione definitiva che SHA-1 non è più sicuro

SHA-2 (2001) - NSA:

Una famiglia di funzioni ancora considerate sicure:

Variante	Output	Stato interno	Sicurezza
SHA-224	224 bit	256 bit	2^{112}
SHA-256	256 bit	256 bit	2^{128}
SHA-384	384 bit	512 bit	2^{192}
SHA-512	512 bit	512 bit	2^{256}

La struttura di SHA-256:

```
1. Padding: aggiungere bit fino a lunghezza ≡ 448 (mod 512)
   Aggiungere lunghezza originale (64 bit)

2. Dividere in blocchi di 512 bit

3. Per ogni blocco:
   - Espandere a 64 word di 32 bit
   - 64 round di operazioni:
     • Funzioni logiche: Ch, Maj
     • Rotazioni: RotR, Shr
     • Addizioni mod 2^32
   - Aggiornare lo stato (8 registri di 32 bit)

4. Concatenare i registri finali → hash di 256 bit
```

Le costanti di SHA-256:

Derivate dalle radici cubiche dei primi 64 numeri primi - nessun numero magico arbitrario!

K[0] = 0x428a2f98 (prima frazione di $\sqrt[3]{2}$)
K[1] = 0x71374491 (prima frazione di $\sqrt[3]{3}$)
...

Lo status attuale: SHA-256 e SHA-512 sono considerati sicuri. Nessun attacco pratico noto.

19.7 SHA-3 (2015) - Keccak

Dopo gli attacchi a MD5 e SHA-1, il NIST lanciò nel 2007 un concorso per un nuovo standard hash.

Il vincitore (2012): Keccak, progettato da Guido Bertoni, Joan Daemen (co-creatore di AES), Michaël Peeters e Gilles Van Assche.

La struttura radicalmente diversa:

SHA-3 usa una costruzione sponge invece di Merkle-Damgård:

1. Stato interno: 1600 bit (matrice 5×5×64)

2. Fase di "absorbing":
   - Input diviso in blocchi
   - Ogni blocco XORato nello stato
   - Applicare la funzione di permutazione Keccak-f

3. Fase di "squeezing":
   - Estrarre bit dallo stato
   - Se serve più output, applicare ancora Keccak-f

La funzione Keccak-f:

24 round di 5 operazioni (θ, ρ, π, χ, ι) che mescolano completamente lo stato.

Le varianti:

Variante	Output	Capacità	Sicurezza
SHA3-224	224 bit	448 bit	2^{112}
SHA3-256	256 bit	512 bit	2^{128}
SHA3-384	384 bit	768 bit	2^{192}
SHA3-512	512 bit	1024 bit	2^{256}

I vantaggi di SHA-3:

- Design completamente diverso da SHA-2 (diversificazione del rischio)
- Più veloce in hardware
- Supporto nativo per output di lunghezza variabile (SHAKE128/256)
- Resistenza teoricamente dimostrata contro molte classi di attacchi

L'adozione:

SHA-3 è pubblicato come standard, ma SHA-2 rimane più comune perché:
- SHA-2 è ancora sicuro
- SHA-2 ha supporto hardware (istruzioni CPU dedicate)
- Cambiare standard richiede tempo e investimenti

CAPITOLO 20

LA CRITTOGRAFIA A CURVA ELLITTICA (ECC)

20.1 Il Problema delle Chiavi Lunghe

L'RSA e il Diffie-Hellman classico hanno un problema crescente: le dimensioni delle chiavi.

Per sicurezza equivalente:

Simmetrico	RSA/DH	Rapporto
80 bit	1024 bit	12,8×
128 bit	3072 bit	24×
256 bit	15360 bit	60×

I problemi pratici:

- Chiavi grandi → più banda necessaria per la trasmissione
- Operazioni lente → calcoli costosi (specialmente su dispositivi mobili)
- Memoria → smartcard e dispositivi IoT hanno RAM limitata

La soluzione: Le curve ellittiche offrono la stessa sicurezza con chiavi molto più corte.

20.2 Le Curve Ellittiche: La Matematica

Una curva ellittica su un campo finito F_p è l'insieme dei punti (x, y) che soddisfano:

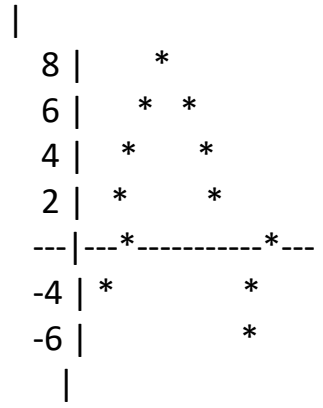
$y^2 = x^3 + ax + b \pmod p$

più un punto speciale O (il "punto all'infinito").

Esempio: La curva $y^2 = x^3 + 7 \pmod{17}$

I punti sulla curva includono:
- (0, 6), (0, 11)
- (1, 5), (1, 12)
- (2, 5), (2, 12)
- ...e altri, più O

La visualizzazione (sui reali):



Su $\mathbb{R}$, la curva $y^2 = x^3 - 3x + 5$ ha una forma caratteristica simmetrica rispetto all'asse x, con due componenti connesse che si avvicinano all'infinito.

20.3 L'Operazione di Gruppo: Addizione di Punti

Possiamo definire un'operazione "+" sui punti della curva che forma un gruppo abeliano.

La regola geometrica (sui reali):

Per sommare P + Q:
1. Tracciare la retta che passa per P e Q
2. Questa retta interseca la curva in un terzo punto R'
3. P + Q = R, dove R è la riflessione di R' rispetto all'asse x

Il caso speciale: raddoppio di un punto (P + P = 2P):

Tracciare la tangente alla curva in P, trovare il terzo punto di intersezione.
Le formule algebriche (su F_p):

Dati $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ sulla curva $y^2 = x^3 + ax + b \pmod p$:

Caso 1: $P \neq Q$ (addizione ordinaria)

Calcolare la pendenza:
$\lambda = (y_2 - y_1) \times (x_2 - x_1)^{-1} \bmod p$

Allora $P + Q = (x_3, y_3)$ dove:
- $x_3 = \lambda^2 - x_1 - x_2 \bmod p$
- $y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$

Caso 2: $P = Q$ (raddoppio)

$\lambda = (3x_1^2 + a) \times (2y_1)^{-1} \bmod p$
- $x_3 = \lambda^2 - 2x_1 \bmod p$
- $y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$

Caso 3: Punto all'infinito

- $P + O = P$ (O è l'elemento neutro)
- $P + (-P) = O$ (dove $-P = (x_1, -y_1)$)

Esempio numerico:

Curva: $y^2 = x^3 + 2x + 3 \pmod{97}$
$P = (3, 6)$, $Q = (80, 10)$

Calcolare $P + Q$:

$\lambda = (10 - 6) \times (80 - 3)^{-1} \bmod 97$
$= 4 \times 77^{-1} \bmod 97$
$= 4 \times 19 \bmod 97$ (perché $77 \times 19 \equiv 1 \bmod 97$)
$= 76 \bmod 97$

$x_3 = 76^2 - 3 - 80 \bmod 97 = 5776 - 83 \bmod 97 = 80 \bmod 97$
$y_3 = 76(3 - 80) - 6 \bmod 97 = -5858 \bmod 97 = 87 \bmod 97$

$P + Q = (80, 87)$ ✓

20.4 La Moltiplicazione Scalare
L'operazione fondamentale nell'ECC è la moltiplicazione scalare:

kP = P + P + ... + P (k volte)

L'algoritmo efficiente: Double-and-Add

Simile all'esponenziazione veloce (square-and-multiply):

Input: k (scalare), P (punto)
Output: kP

1. Scrivere k in binario: k = k_n k_{n-1} ... k_1 k_0
2. Q ← O (punto all'infinito)
3. Per i da n a 0:
  Q ← 2Q (raddoppio)
  Se k_i = 1:
    Q ← Q + P (addizione)
4. Return Q

La complessità: O(log k) operazioni invece di O(k).

Esempio:

Calcolare 23P dove $23 = 10111_2$

Bit corrente	Operazione	Risultato
1 (MSB)	Q = P	P
0	Q = 2P	2P
1	Q = 4P + P	5P
1	Q = 10P + P	11P
1	Q = 22P + P	23P

```

## 20.5 Il Problema del Logaritmo Discreto su Curve Ellittiche (ECDLP)

Dato un punto base G sulla curva e un punto Q = kG, trovare lo scalare k.

La difficoltà:

Il miglior algoritmo noto ha complessità $O(\sqrt{n})$ dove n è l'ordine del gruppo (il numero di punti sulla curva).

Per una curva con $n \approx 2^{256}$ punti, serve circa $2^{128}$ operazioni - computazionalmente impraticabile!
```

Il confronto con RSA:

- RSA 3072 bit ≈ sicurezza 128 bit (fattorizzazione sub-esponenziale)
- ECC 256 bit ≈ sicurezza 128 bit (ECDLP esponenziale)

L'ECC è molto più efficiente per la stessa sicurezza.

20.6 ECDH: Diffie-Hellman su Curve Ellittiche

Il protocollo:

Setup pubblico:
- Una curva ellittica E definita su F_p
- Un punto base G sulla curva di ordine n

Alice e Bob:

Alice:
- Sceglie uno scalare segreto a ∈ [1, n-1]
- Calcola A = aG
- Invia A a Bob

Bob:
- Sceglie uno scalare segreto b ∈ [1, n-1]
- Calcola B = bG
- Invia B ad Alice

Chiave condivisa:

Alice calcola: S = aB = a(bG) = (ab)G
Bob calcola: S = bA = b(aG) = (ab)G

Entrambi ottengono lo stesso punto S!

Eva vede: E, p, G, n, A = aG, B = bG

Ma non può calcolare S = (ab)G senza conoscere a o b (ECDLP).

Esempio pratico (con numeri piccoli):
Curva: $y^2 = x^3 + 2x + 3 \pmod{97}$
G = (3, 6), ordine n = 5

Alice sceglie a = 2
A = 2G = G + G = (3,6) + (3,6) = (80, 10)
Alice invia A = (80, 10)

Bob sceglie b = 3
B = 3G = 2G + G = (80, 10) + (3, 6) = (80, 87)
Bob invia B = (80, 87)

Chiave condivisa:
Alice: S = 2B = 2(80, 87) = ... = (3, 91)
Bob: S = 3A = 3(80, 10) = ... = (3, 91)

S = (3, 91) ✓

20.7 ECDSA: Firme Digitali con Curve Ellittiche

La generazione delle chiavi:

- Scegliere uno scalare privato $d \in [1, n-1]$
- Calcolare $Q = dG$ (chiave pubblica)

La firma di un messaggio m:

1. Calcolare $e = H(m)$ (hash del messaggio)
2. Scegliere un nonce casuale $k \in [1, n-1]$
3. Calcolare $(x_1, y_1) = kG$
4. Calcolare $r = x_1 \bmod n$ (Se $r = 0$, scegliere un altro k)
5. Calcolare $s = k^{-1}(e + dr) \bmod n$ (Se $s = 0$, scegliere un altro k)
6. La firma è (r, s)

La verifica della firma (r, s) per il messaggio m:

1. Verificare che $r, s \in [1, n-1]$
2. Calcolare $e = H(m)$
3. Calcolare $w = s^{-1} \bmod n$
4. Calcolare $u_1 = ew \bmod n$ e $u_2 = rw \bmod n$
5. Calcolare $(x_1, y_1) = u_1G + u_2Q$
6. La firma è valida se e solo se $x_1 \equiv r \pmod n$

La dimostrazione della correttezza:

$u_1G + u_2Q = u_1G + u_2(dG) = (u_1 + u_2d)G$

$u_1 + u_2d = ew + rwd = w(e + rd) = s^{-1}(e + rd)$
Per definizione: $s = k^{-1}(e + dr)$, quindi $s^{-1}(e + rd) = k$

Quindi: $u_1G + u_2Q = kG$

Che è esattamente il punto usato per generare r! ✓

20.8 Le Curve Standard: P-256, secp256k1, Curve25519

NIST P-256 (secp256r1):

La curva standardizzata dal NIST per uso governativo USA.
```
p = 2^256 - 2^224 + 2^192 + 2^96 - 1
a = -3
b = 41058363725152142129326129780047268409114441015993725554835256314039467401291
G = (punto base specifico)
n = (ordine del gruppo, ≈ 2^256)
```

Usata in: TLS, IPsec, SSH, PGP/GPG

secp256k1:

La curva usata da Bitcoin e molte altre criptovalute.

```
p = 2^256 - 2^32 - 977
a = 0
b = 7
Equazione: y² = x³ + 7 (molto semplice!)
```

Le proprietà speciali:

- La forma di Koblitz permette calcoli più efficienti
- Parametri scelti in modo "nothing-up-my-sleeve" (niente-di-nascosto)

Curve25519:

Progettata da Daniel J. Bernstein nel 2005 per massima sicurezza ed efficienza.

Montgomery curve: $y^2 = x^3 + 486662x^2 + x \pmod{2^{255} - 19}$

I vantaggi:

- Estremamente veloce (ottimizzata per CPU moderne)
- Resistente a molte classi di attacchi side-channel
- Design pubblico e verificabile
- Nessuna scelta arbitraria di parametri

L'uso: Signal, WhatsApp, Tor, SSH (algoritmo X25519)

Ed25519: Variante per firme digitali (usata da GitHub, OpenSSH, ecc.)

20.9 I Vantaggi dell'ECC

Confronto pratico:

Sicurezza	Simmetrico	RSA	ECC	Rapporto
80 bit	AES-80	1024	160	6,4×
128 bit	AES-128	3072	256	12×
192 bit	AES-192	7680	384	20×
256 bit	AES-256	15360	512	30×

Le implicazioni:

1. Dimensioni delle chiavi più piccole:
   - Certificati SSL più compatti
   - Meno banda per la trasmissione
   - Più adatto per dispositivi mobili/IoT

2. Calcoli più veloci:
   - Firma/verifica più rapida
   - Meno consumo energetico (importante per batterie)

3. Memoria ridotta:
   - Fattibile su smartcard con RAM limitata
   - Più operazioni concorrenti possibili

Le limitazioni:

- Implementazione più complessa (matematica delle curve)
- Vulnerabilità agli attacchi side-channel se non implementato correttamente
- Potenzialmente vulnerabile ai computer quantistici (come RSA)

CAPITOLO 21

LA MINACCIA QUANTISTICA E LA CRITTOGRAFIA POST-QUANTISTICA

21.1 I Computer Quantistici: Una Nuova Minaccia

Un computer quantistico sfrutta le proprietà della meccanica quantistica (sovrapposizione, entanglement) per eseguire certi calcoli in modo radicalmente più efficiente dei computer classici.

Lo stato attuale (2024):

- I computer quantistici esistono ma sono ancora limitati
- Pochi qubit (unità quantistiche di informazione)
- Alto tasso di errore
- Richiedono temperature vicine allo zero assoluto
- Molto costosi

Il progresso:

Anno	Qubit	Organizzazione
2019	53	Google (Sycamore)
2021	127	IBM (Eagle)
2022	433	IBM (Osprey)
2023	1121	IBM (Condor)

La proiezione: Computer quantistici pratici per la crittanalisi potrebbero esistere tra 10-20 anni.

21.2 L'Algoritmo di Shor: La Fine di RSA ed ECC

Nel 1994, il matematico Peter Shor scoprì un algoritmo quantistico che risolve due problemi fondamentali:
1. La fattorizzazione:

Fattorizzare $n = pq$ in tempo $O((\log n)^3)$ con un computer quantistico.

2. Il logaritmo discreto:

Risolvere g^x ≡ y (mod p) in tempo O((log p)³).

Il confronto con gli algoritmi classici:

Problema	Classico	Quantistico (Shor)
Fattorizzare 2048 bit	~10^9 anni	~ore
DLP 2048 bit	~10^9 anni	~ore
ECDLP 256 bit	~10^{38} anni	~minuti

Ecco la descrizione tecnica dell'algoritmo:

Obiettivo:

Dato un numero intero composito 푁, trovare due fattori primi 푝 e 푞 tali che:

N = p * q

Riduzione classica del problema

Scegli un intero 푎 tale che:

1 < a < N e MCD(a, N) = 1

Definisci la funzione periodica:

f(x) = a^x mod N

L'obiettivo è trovare il periodo 푟 di 푓(푥), cioè il più piccolo intero positivo tale che:

a^r ≡ 1 mod N

Parte quantistica: calcolo del periodo 푟

Preparazione dello stato quantistico:

Si usano due registri quantistici:
Primo registro:

|푞⟩, con 푞∈ {0,1,...,푁−1}, dove

푁= 2푛 e 푁> 푟^2

Secondo registro: inizializzato a

|0⟩

Stato iniziale:

|ψ_0⟩ = (1/√Q) ∑_{x=0}^{Q-1} |x⟩|0⟩

Valutazione della funzione:

Applica l'operatore unitario 푈푓 che calcola 푓(푞) = 푎^푞 푚표푑 푁

|ψ_1⟩ = (1/√Q) ∑_{x=0}^{Q-1} |x⟩|f(x)⟩

Misura il secondo registro:

Collassa lo stato sul valore 푓(푞0), lasciando una sovrapposizione coerente degli 푞 tali che

푓(푞) = 푓(푞0)

|ψ_2⟩ = (1/√K) ∑_{k=0}^{K-1} |x_0 + k*r⟩

Applica la Trasformata di Fourier Quantistica (QFT) al primo registro:

La QFT trasforma la periodicità in picchi nella frequenza:

QFT(|ψ_2⟩) = ∑_{y=0}^{Q-1} c_y |y⟩

I coefficienti 푐푦 sono massimi per 푦≈ 푁∗ 푗/푟, con 푗∈ 푍

Parte classica finale
Verifica che 푟 sia pari e che:

a^{r/2} ≠ -1 mod N

Calcola i fattori:

p = MCD(a^{r/2} - 1, N)

q = MCD(a^{r/2} + 1, N)

Se 푞 e 푟 sono non banali (diversi da 1 e 푁), allora:

N = p * q

Complessità
L'algoritmo di Shor ha complessità temporale:

O((log N)^3)
contro l'approccio classico più efficiente noto (algoritmo di Lenstra o di Dixon) che è sub-esponenziale.

L'implicazione devastante. Quando i computer quantistici diventeranno pratici:

- RSA: Completamente rotto
- Diffie-Hellman: Completamente rotto
- ECC (ECDH, ECDSA): Completamente rotto
- DSA: Completamente rotto

L'intera infrastruttura di sicurezza moderna (TLS, blockchain, firme digitali) diventa insicura!

21.3 L'Algoritmo di Grover: Minaccia alla Crittografia Simmetrica

Nel 1996, Lov Grover scoprì un altro algoritmo quantistico:

Il problema: Ricerca in un database non ordinato di N elementi.

Classico: O(N) (devi guardare ogni elemento)
Quantistico (Grover): O(√N)

L'applicazione alla crittografia:

Per rompere una chiave simmetrica di n bit con forza bruta:

- Classico: 2^n tentativi

- Quantistico: 2^(n/2) tentativi

Le implicazioni:

Algoritmo	Chiave classica	Sicurezza effettiva
AES-128	128 bit	64 bit (insicuro)
AES-192	192 bit	96 bit (marginale)
AES-256	256 bit	128 bit (sicuro)

La soluzione: Raddoppiare le dimensioni delle chiavi simmetriche.

- AES-128 → AES-256
- SHA-256 → SHA-512

21.4 La Risposta: La Crittografia Post-Quantistica

Nel 2016, il NIST lanciò un processo di standardizzazione per algoritmi resistenti ai computer quantistici.

I requisiti:

1. Sicurezza contro attacchi classici e quantistici
2. Efficienza pratica (non troppo lento)
3. Dimensioni ragionevoli di chiavi/firme
4. Basato su problemi matematici diversi da fattorizzazione e logaritmo discreto

Le categorie di approcci:

21.4.1 I Reticoli (Lattice-based)

Il problema matematico: Shortest Vector Problem (SVP) e Learning With Errors (LWE).

Dato un reticolo (un insieme discreto di punti nello spazio), trovare il vettore più corto.

Il problema LWE:

Dato un sistema di equazioni lineari con rumore:

$Ax + e = b \pmod q$

dove e è un piccolo vettore di errore, trovare x.

Questo problema è considerato difficile anche per computer quantistici.

I candidati principali:

CRYSTALS-Kyber (vincitore NIST 2022):

- Scambio di chiavi (KEM - Key Encapsulation Mechanism)
- Chiave pubblica: circa 800-1200 byte
- Testo cifrato: circa 700-1100 byte
- Veloce e efficiente

CRYSTALS-Dilithium (vincitore NIST 2022):

- Firme digitali
- Chiave pubblica: circa 1300 byte
- Firma: circa 2400 byte
- Veloce nella firma e verifica

21.4.2 I Codici (Code-based)

Il problema matematico: Syndrome Decoding Problem.

Decodificare un codice lineare casuale è NP-completo.

Classic McEliece (finalista NIST):

Basato sui codici di Goppa, proposti già nel 1978 da Robert McEliece.

I vantaggi:

- Sicurezza ben compresa (45+ anni di analisi)
- Veloce nella cifratura/decifratura

Lo svantaggio:

- Chiavi pubbliche enormi: 200 KB - 1 MB!
- Impraticabile per molte applicazioni

21.4.3 Gli Hash (Hash-based)

Il principio: Costruire firme digitali usando solo funzioni di hash crittografiche.

SPHINCS+ (finalista NIST):

- Basato su hash trees (alberi di Merkle)
- Chiave pubblica piccola: circa 32-64 byte
- Firma grande: circa 8-50 KB
- Lento ma molto sicuro

I vantaggi:

- Sicurezza basata solo sulla robustezza delle funzioni hash
- Minimo set di assunzioni

Gli svantaggi:

- Firme molto grandi
- Lento nella generazione delle firme

21.4.4 Gli Isogeni (Isogeny-based)

Il problema matematico: Trovare isogenie tra curve ellittiche.

SIKE (candidato NIST - ROTTO 2022!):

Basato su cammini in grafi di isogenie di curve ellittiche supersingolari.

Il vantaggio: Chiavi molto piccole (circa 300 byte).

La beffa: Nel luglio 2022, Castryck e Decru pubblicarono un attacco che rompe SIKE in poche ore su un PC!
Questo dimostra quanto sia difficile progettare crittografia post-quantistica sicura.

21.5 La Timeline della Transizione

2016: NIST avvia il processo di standardizzazione

2022: NIST annuncia i vincitori:
- KEM: CRYSTALS-Kyber
- Firme digitali: CRYSTALS-Dilithium, FALCON, SPHINCS+

2024: Standard finali pubblicati (FIPS 203, 204, 205)

2025-2030: Periodo di transizione
- Implementazione nei protocolli (TLS 1.3, ecc.)
- Aggiornamento software/firmware
- Certificati ibridi (classici + post-quantistici)

2030-2035: Deprecazione degli algoritmi classici
- RSA, ECDSA gradualmente eliminati
- Migrazione completa al post-quantistico

Il problema della "harvest now, decrypt later":

Gli attaccanti possono:
1. Registrare oggi tutto il traffico cifrato (anche se non possono decifrarlo)
2. Aspettare che i computer quantistici diventino disponibili
3. Decifrare retroattivamente i dati registrati

Le informazioni sensibili a lungo termine (dati medici, segreti governativi, proprietà intellettuale) sono già a rischio!

La transizione deve avvenire ora, non quando i computer quantistici saranno realtà.

21.6 Gli Algoritmi Ibridi: Il Ponte verso il Futuro

Durante la transizione, si usano schemi ibridi che combinano algoritmi classici e post-quantistici:

Esempio per TLS:

```
1. Handshake classico: ECDH (X25519)
2. Handshake PQC: Kyber
3. Chiave finale: KDF(chiave_ECDH  chiave_Kyber)
```

I vantaggi:

- Se Kyber viene rotto, abbiamo ancora la sicurezza di X25519 (contro attaccanti classici)
- Se X25519 viene rotto da computer quantistici, abbiamo Kyber
- Protezione massima durante la transizione

Gli svantaggi:

- Handshake più lento
- Più dati trasmessi (chiavi pubbliche più grandi)
- Complessità implementativa

CAPITOLO 22

LA CRITTOGRAFIA APPLICATA E IL FUTURO

22.1 TLS/SSL: La Sicurezza del Web

Transport Layer Security (TLS) è il protocollo che protegge quasi tutto il traffico Internet moderno.

La storia:

- 1994: SSL 1.0 (Netscape, mai rilasciato)
- 1995: SSL 2.0 (vulnerabilità severe)
- 1996: SSL 3.0 (migliorato ma ancora vulnerabile)
- 1999: TLS 1.0 (standardizzato, basato su SSL 3.0)
- 2006: TLS 1.1
- 2008: TLS 1.2 (ancora ampiamente usato)
- 2018: TLS 1.3 (versione moderna, più veloce e sicura)

L'handshake TLS 1.3 (semplificato):

```
Client → Server:
  ClientHello
    - Versioni TLS supportate
    - Cipher suite supportate (es. AES-256-GCM)
    - Chiavi pubbliche per key exchange (X25519, Kyber, ecc.)
    - Estensioni (SNI, ALPN, ecc.)

Server → Client:
  ServerHello
    - Versione TLS scelta
    - Cipher suite scelta
    - Chiave pubblica per key exchange
  {Certificato del server}
  {CertificateVerify}
  {Finished}

Client → Server:
```

```
  {Certificato del client} (opzionale)
  {CertificateVerify}
  {Finished}
 = cifrato con le chiavi derivate dallo scambio

↓
Dati applicativi cifrati
```
Le cipher suite moderne:
```
TLS_AES_256_GCM_SHA384
└┬┘ └─┬─┘ └┬┘ └──┬──┘
 │   │    │    └─ Hash: SHA-384
 │   │    └──────── Modalità: GCM (Galois/Counter Mode)
 │   └───────────── Cifrario: AES-256
 └──────────────────── Protocollo: TLS
```

I miglioramenti in TLS 1.3:

1. 1-RTT handshake: Una sola andata-ritorno invece di due
2. 0-RTT resumption: Riconnessione istantanea (con limitazioni di sicurezza)
3. Forward secrecy obbligatoria: Usa sempre ECDHE/DHE
4. Cipher suite semplificate: Rimosse opzioni insicure/obsolete
5. Cifratura del certificato: Il certificato del server è sempre cifrato

22.2 La Forward Secrecy (Perfect Forward Secrecy - PFS)

Il problema con RSA puro:

```
ClientKeyExchange: RSA_encrypt(PubKey_Server, pre_master_secret)
```

Se la chiave privata del server viene compromessa in futuro, un attaccante che ha registrato tutto il traffico passato può:

1. Decifrare tutti i ClientKeyExchange registrati
2. Ricavare tutti i pre_master_secret
3. Derivare tutte le chiavi di sessione
4. Decifrare tutto il traffico passato!

La soluzione: Diffie-Hellman Effimero (DHE/ECDHE):

Per ogni sessione:
1. Il server genera nuova coppia di chiavi DH effimere (valide solo per quella sessione)
2. Scambio DH con il client
3. La chiave di sessione deriva dallo scambio DH
4. Dopo la sessione, le chiavi effimere vengono eliminate
```
Il risultato:

Anche se la chiave privata del server (usata per firmare, non cifrare) viene compromessa, le sessioni passate rimangono sicure perché:

- Le chiavi effimere sono state distrutte
- Non c'è modo di ricalcolarle
- Ogni sessione è indipendente

L'adozione:

TLS 1.3 richiede forward secrecy - non supporta più RSA key transport.

## 22.3 I Certificati e la PKI (Public Key Infrastructure)

Il problema della distribuzione delle chiavi pubbliche:

Come fa Bob a sapere che la chiave pubblica che ha ricevuto appartiene veramente ad Alice e non a un attaccante?

La soluzione: Certificate Authorities (CA):

Un'autorità fidata che:

1. Verifica l'identità del richiedente
2. Firma digitalmente il certificato che lega identità ↔ chiave pubblica

La struttura di un certificato X.509:

```
Certificato:
  Versione: 3
  Serial Number: 04:de:1b:1e:...

  Subject (proprietario):
    CN (Common Name): www.example.com
```
```

```
    O (Organization): Example Inc.
    C (Country): US

  Issuer (emittente/CA):
    CN: DigiCert Global Root CA
    O: DigiCert Inc
    C: US

  Validità:
    Not Before: 2024-01-01 00:00:00
    Not After: 2025-01-01 23:59:59

  Chiave pubblica:
    Algorithm: RSA 2048 bit
    Public Key: 30 82 01 0a 02 82 01 01 00 ...

  Estensioni:
    Subject Alternative Names: example.com, .example.com
    Key Usage: Digital Signature, Key Encipherment
    Extended Key Usage: Server Authentication

  Firma della CA:
    Algorithm: sha256WithRSAEncryption
    Signature: 3d fd 24 e2 ...
```

La catena di fiducia:

```
Root CA (auto-firmato, nel browser)
  └─ Intermediate CA (firmato da Root)
      └─ Server Certificate (firmato da Intermediate)
```

La verifica:

1. Browser riceve il certificato del server
2. Verifica la firma dell'Intermediate CA
3. Verifica la firma della Root CA (già fidato)
4. Verifica che il certificato non sia scaduto/revocato
5. Verifica che il CN corrisponda al dominio visitato

La revoca dei certificati:

Se una chiave privata viene compromessa, il certificato deve essere revocato:

- CRL (Certificate Revocation List): Lista di certificati revocati, pubblicata periodicamente dalla CA
- OCSP (Online Certificate Status Protocol): Check in tempo reale dello stato del certificato
- OCSP Stapling: Il server include la risposta OCSP nell'handshake TLS

22.4 La Crittografia End-to-End: Il Signal Protocol

Il Signal Protocol (precedentemente Axolotl) è considerato il gold standard per la messaggistica sicura.

Usato da:

- Signal (ovviamente)
- WhatsApp (2 miliardi di utenti)
- Facebook Messenger (modalità "secret conversations")
- Google Messages (RCS con E2EE)
- Skype (conversazioni private)

Le proprietà chiave:

1. End-to-End Encryption (E2EE): Solo mittente e destinatario possono leggere i messaggi, nemmeno il server
2. Perfect Forward Secrecy (PFS): La compromissione delle chiavi a lungo termine non compromette le sessioni passate
3. Future Secrecy (Break-in Recovery): La compromissione di una chiave di sessione non compromette le sessioni future
4. Deniability: Non si può provare crittograficamente chi ha scritto un messaggio (a differenza delle firme digitali)

L'architettura del Signal Protocol:

1. Le chiavi a lungo termine:

- Identity Key Pair (IK): Coppia di chiavi Curve25519, generata all'installazione, identifica il dispositivo
- Signed PreKey (SPK): Chiave firmata con IK, cambiata periodicamente (es. settimanalmente)
- One-Time PreKeys (OPK): Batch di chiavi effimere caricate sul server

2. Il X3DH (Extended Triple Diffie-Hellman) handshake:

Quando Alice vuole iniziare una conversazione con Bob:

```
Alice conosce:
- La propria IK_a (privata)
- IK_b, SPK_b, OPK_b di Bob (scaricate dal server)

Alice genera:
- Una chiave effimera EK_a

Alice calcola 4 scambi DH:
DH1 = DH(IK_a, SPK_b)
DH2 = DH(EK_a, IK_b)
DH3 = DH(EK_a, SPK_b)
DH4 = DH(EK_a, OPK_b) [se disponibile]

Chiave iniziale:
SK = KDF(DH1  DH2  DH3  DH4)
```

L'intuizione: Quattro scambi DH indipendenti assicurano che la compromissione di una chiave non comprometta tutto. Anche se l'attaccante controlla il server, non può impersonare Bob senza la sua chiave privata.

3. Il Double Ratchet Algorithm:

Dopo l'handshake iniziale, il Double Ratchet gestisce le chiavi di sessione.

Il Symmetric-Key Ratchet (Chain Key Ratchet):

```
Per ogni messaggio:
1. Chain Key (CK) deriva la prossima:
   Message Key (MK) = KDF(CK, 0x01)
   CK_next = KDF(CK, 0x02)

2. Usa MK per cifrare il messaggio (AES-CBC + HMAC)
3. Elimina MK immediatamente dopo l'uso
4. Aggiorna CK ← CK_next
```

Ogni messaggio usa una chiave diversa. Se una chiave viene compromessa, solo quel messaggio è esposto.

Il Diffie-Hellman Ratchet:

Ogni volta che Alice riceve un messaggio da Bob:
1. Bob invia la sua nuova chiave DH pubblica
2. Alice genera una nuova coppia DH
3. Calcolano un nuovo shared secret
4. Derivano nuove Chain Keys da questo secret
```

Questo assicura forward secrecy e future secrecy: anche se un attaccante ottiene le chiavi attuali, non può leggere messaggi futuri (dopo il prossimo ratchet DH).

Il funzionamento combinato:

```
Alice → Bob: msg1 (usando MK1 derivata da CK)
Alice → Bob: msg2 (usando MK2 derivata da CK)
      ↓
Bob risponde: [nuova chiave DH pubblica]
      ↓
[DH Ratchet: nuovo shared secret, nuove CK]
      ↓
Bob →  Alice: msg3 (usando nuova MK3)
Alice → Bob: msg4 (usando nuova MK4)
```

I metadati protetti:

Signal nasconde anche i metadati usando:

- Sealed Sender: Il server non sa chi sta inviando a chi
- Private Contact Discovery: Trovare contatti senza rivelare la rubrica completa
- Padding dei messaggi: Nascondere la lunghezza effettiva

22.5 La Blockchain e le Criptovalute

Bitcoin (2008): La prima applicazione su larga scala della crittografia moderna per creare moneta digitale decentralizzata.

I componenti crittografici di Bitcoin:

1. Le firme digitali (ECDSA su secp256k1):
Ogni transazione è firmata digitalmente:
```

Alice possiede:

- Chiave privata: d_a
- Chiave pubblica: Q_a = d_a × G
- Indirizzo Bitcoin: Hash(Q_a)

Per spendere:
1. Alice crea una transazione: "Paga 1 BTC da indirizzo_Alice a indirizzo_Bob"
2. Firma con ECDSA usando d_a
3. Broadcast alla rete

Verifica:
- Chiunque può verificare la firma usando Q_a
- Solo Alice può creare firme valide (possiede d_a)
```

2. Le funzioni di hash (SHA-256, RIPEMD-160):

- Indirizzi: Derivati da Hash(chiave_pubblica)
- Merkle Trees: Struttura efficiente per memorizzare transazioni
- Proof-of-Work: Mining richiede trovare hash con proprietà specifiche

3. Il Proof-of-Work (Mining):

Problema: Trovare un nonce tale che:
SHA-256(SHA-256(header_blocco  nonce)) < target

dove target è un numero molto piccolo
(es. richiede 19 zeri iniziali in esadecimale)

Soluzione: Provare miliardi di nonce fino a trovarne uno valido
```
La difficoltà: Regolata ogni 2016 blocchi (circa 2 settimane) per mantenere un tempo medio di 10 minuti per blocco.

Il consumo energetico: La rete Bitcoin consuma circa 150 TWh/anno (più dell'Argentina) per il mining.

Ethereum 2.0 (Proof-of-Stake): Risolve il problema energetico sostituendo il mining con lo staking, usando firme aggregate BLS.

22.6 La Privacy Differenziale

Il concetto: Come condividere statistiche su un dataset proteggendo la privacy dei singoli individui?

La definizione formale:

Un meccanismo M soddisfa ε-differential privacy se per tutti i dataset D_1 e D_2 che differiscono per un solo record, e per tutti gli output S:

$P[M(D_1) \in S] \leq e^{\varepsilon} \times P[M(D_2) \in S]$

L'interpretazione: L'inclusione o esclusione di un singolo record cambia la distribuzione dell'output di al massimo un fattore e^{ε}.

L'implementazione pratica: aggiungere rumore:

Query: "Quante persone hanno età > 50?"
Risposta vera: 1247

Privacy differenziale:
1. Calcola la risposta vera: 1247
2. Aggiungi rumore Laplaciano: 1247 + Lap(0, Δf/ε)
3. Rispondi: 1253 (ad esempio)
```
Il rumore protegge i singoli individui ma permette analisi statistiche accurate su grandi dataset.

L'uso in pratica:

- Apple: Raccolta dati iOS con differential privacy
- Google: Chrome, statistiche di utilizzo
- US Census Bureau: Censimento 2020

## 22.7 La Crittografia Omomorfica

L'idea rivoluzionaria: Eseguire calcoli su dati cifrati senza doverli decifrare.

La proprietà omomorfica:

Se E è la funzione di cifratura, vogliamo che:
$E(a) \oplus E(b) = E(a \oplus b)$

per qualche operazione $\oplus$.

Esempi:
```

1. Crittografia omomorfica additiva (RSA parzialmente omomorfo):

```
E(m₁) × E(m₂) mod n = E(m₁ + m₂ mod ϕ(n))
```

2. Fully Homomorphic Encryption (FHE):

Supporta sia addizione che moltiplicazione sui testi cifrati, permettendo calcoli arbitrari.

Esempio pratico:

```
Cloud computing sicuro:
1. Alice cifra i dati: C = E(dati)
2. Invia C al cloud
3. Cloud esegue calcoli: Risultato_cifrato = F(C)
4. Cloud restituisce Risultato_cifrato
5. Alice decifra: risultato = D(Risultato_cifrato)

Il cloud non vede mai i dati in chiaro!
```

I candidati FHE moderni:
- BGV (Brakerski-Gentry-Vaikuntanathan)
- BFV (Brakerski-Fan-Vercauteren)
- CKKS (Cheon-Kim-Kim-Song): Ottimizzato per calcoli approssimati (machine learning)

Il problema: Ancora molto lento (1000-10000× più lento dei calcoli normali) e richiede chiavi molto grandi.

Le applicazioni emergenti:
- Machine learning su dati sensibili (medici, finanziari)
- Votazione elettronica
- Ricerca su database cifrati

22.8 Zero-Knowledge Proofs (ZKP)

Il concetto: Provare di conoscere un segreto senza rivelarlo.

L'esempio classico: La caverna di Ali Baba:

```
Scenario:
- Una caverna con due percorsi (A e B) che si incontrano
- Una porta magica al punto di incontro (serve password per aprire)
- Alice conosce la password, vuole provarlo a Bob senza rivelarla

Protocollo:
1. Alice entra nella caverna (Bob non vede quale percorso sceglie)
2. Bob arriva all'entrata e chiede: "Esci dal percorso A" (o B, casuale)
3. Alice usa la password per tornare dal percorso richiesto
4. Ripetere n volte

Se Alice non conosce la password:
- Probabilità di successo in un round: 50%
- Probabilità di successo in n round: (1/2)ⁿ

Con n = 20: probabilità di imbroglio < 0,0001%
```

Le proprietà richieste:

1. Completeness: Se l'affermazione è vera, il verificatore onesto sarà convinto
2. Soundness: Se l'affermazione è falsa, nessun prover disonesto può convincere il verificatore (tranne con probabilità trascurabile)
3. Zero-Knowledge: Il verificatore non apprende nulla oltre al fatto che l'affermazione è vera

ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive ARguments of Knowledge):

Prove molto compatte e verificabili velocemente, senza interazione.

L'uso in Zcash:

```
Transazione privata:
1. Provo di conoscere la chiave privata per spendere le monete
2. Provo che il bilancio è corretto (non creo moneta dal nulla)
3. Ma non rivelo: mittente, destinatario, importo

ZK-SNARK: "Questa transazione è valida" ✓
Dimensione della prova: ~200 byte
```

Le applicazioni moderne:

- Criptovalute: Zcash, Monero (privacy transazioni)
- Layer 2 Scaling: zkRollups su Ethereum
- Autenticazione: Provare identità senza rivelare informazioni
- Compliance: Provare a soddisfare regolamenti senza rivelare dati sensibili

22.9 La Crittografia Quantistica (QKD)

Quantum Key Distribution (QKD): Usare le leggi della fisica quantistica per distribuire chiavi con sicurezza provabile.

Il protocollo BB84 (Bennett-Brassard 1984):

Alice e Bob vogliono condividere una chiave segreta.

1. Alice genera bit casuali e li codifica in stati quantici (fotoni):
   - Bit 0: →⟩ o ↑⟩
   - Bit 1: ↗⟩ o ↖⟩
   Base + : {→⟩, ↑⟩}
   Base × : {↗⟩, ↖⟩}

2. Alice invia i fotoni a Bob

3. Bob misura ogni fotone in base casuale (+ o ×)

4. Alice e Bob confrontano pubblicamente quali basi hanno usato
   (ma non i risultati!)

5. Tengono solo i bit dove hanno usato la stessa base

6. Verificano un campione per rilevare intercettazioni
```
La sicurezza quantistica:

Se Eva intercetta un fotone e lo misura:
- Il principio di indeterminazione di Heisenberg garantisce che la misura disturba lo stato
- Alice e Bob rilevano l'interferenza confrontando un campione
- Abortiscono lo scambio se c'è intercettazione

Lo stato attuale:
```

- Implementazioni commerciali esistono (ID Quantique, Toshiba)
- Distanza limitata (~200 km su fibra ottica)
- Satelliti per QKD a lunga distanza (satellite China's Micius , 2016)

La limitazione: QKD risolve solo la distribuzione delle chiavi, non fornisce crittografia dei dati. Deve essere combinato con cifrari simmetrici (es. AES).

22.10 Il Futuro della Crittografia

Le tendenze emergenti:

1. La crittografia post-quantistica diventerà standard:

- TLS 1.4 con algoritmi PQC
- Certificati ibridi classici+PQC
- Migrazione completa entro il 2035

2. La privacy by design:

- Differential privacy integrata nei sistemi
- Zero-knowledge proofs per compliance
- Minimizzazione dei dati per default

3. L'hardware crittografico:

- Trusted Execution Environments (TEE): Intel SGX, ARM TrustZone
- Hardware Security Modules (HSM) più accessibili
- Crittografia accelerata in GPU e AI accelerators

4. La crittografia leggera (Lightweight Cryptography):

- Algoritmi ottimizzati per IoT con risorse limitate
- NIST Lightweight Cryptography Standard (2023): Ascon
- Protocolli a basso consumo energetico